DOCUMENT RESUME

ED 297 691 IR 012 189

TITLE Artificial Intelligence.
INSTITUTION Harvard Univ., Cambridge, MA. Office of Information Technology.
PUB DATE 85
NOTE 41p.
PUB TYPE Collected Works - Serials (022) -- Historical Materials (060) -- Information Analyses (070)
JOURNAL CIT Information Technology Quarterly; v4 n4 Win 1985

EDRS PRICE MF01/PC02 Plus Postage.
DESCRIPTORS *Artificial Intelligence; Cognitive Processes; *Computer Science; *Expert Systems; History; Intelligence; Law Schools; Lawyers; Logical Thinking; Programing Languages; Robotics; *Technological Advancement

ABSTRACT

This issue of "Information Technology Quarterly" is devoted to the theme of "Artificial Intelligence." It contains two major articles: (1) Artificial Intelligence and Law" (D. Peter O'Neill and George D. Wood); (2) "Artificial Intelligence: A Long and Winding Road" (John J. Simon, Jr.). In addition, it contains two sidebars: (1) "Calculating and Searching"; (2) "Cognitive Science and Artificial Intelligence (Donald S. Bradshaw). The issue is designed to provide a historical perspective as well as to describe some specific applications of artificial intelligence and expert systems. Issues addressed in the first article include a brief history of law and artificial intelligence; efforts to apply artificial intelligence to the field of law; artificial intelligence and legal reasoning; research into the cognitive processes of legal reasoning; Project PERICLES, a joint research endeavor between the Harvard Law School and Digital Equipment Corporation to study the uses of computer technology in the legal domain of landlord tenant law; and a description of cognition and law research. An expert system developed at Harvard (NOMOS--from the Greek concept of law) is described, including implementation methods, modules, and system integration. The second article provides a history of the mechanization of human thought, citing the work of Plato, Descartes, Pascal, Charles Babbage, H. P. Babbage, Alan Turing, Claude Shannon, and others who laid the ground work for today's principles of artificial intelligence. More recent research in the field by American scientists is also discussed, e.g., Alan Newell, Herbert Simon, J. C. Shaw, John McCarthy, Marvin Minsky, Roger Schank, and Robert Wilensky. Several expert systems are also described. Each article includes its own list of references. (JB)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

by 4·16·86

Harvard University

Information Technology

QUARTERLY

Winter 1985
Vol. IV, No. 4

ED297691

Inside this Issue:

Features

Artificial intelligence: a long and winding road / 14
Calculating and searching / 20
Cognitive science and artificial intelligence / 38
Donald S. Bradshaw

Departments

Newsworthy
• Artificial intelligence and law / 3
D. Peter O'Neil and George D. Wood
Information Technology Calendar / 40 *(back cover)*

U.S. DEPARTMENT OF EDUCATION
OERI
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)
☒ This document has been reproduced as received from the person or organization originating it.
☐ Minor changes have been made to improve reproduction quality.
• Points of view or opinions stated in this document do not necessarily represent official position or policy.

ED BY M.A.L. LANE, YOUTH'S COMPANION SERIES, (BOSTON: GINN & COMPANY, THE ATHENAEUM PRESS, 1901), 47.

Prior to the systematic planting of trees in the late nineteenth century, the Landes, a region of France on the Bay of Biscay, south and west of Bordeaux, was characterized by shifting sands, scattered grass and shrubs, and a subsoil of clay that caused rainfall to gather in pools. Shepherds who tended cattle and sheep scattered across these dreary wastes were aided by the expert employment of stilts, upon which they became so adept as to be able to run, hop, and even pick a flower without dismounting. A long pole aided them in mounting, in guiding their animals, and, extended like a third leg of a giant stool, in sitting while they did their knitting. Artificial intelligence research has yielded computer programs, called "expert systems," that function as stilts for the mind, carrying it briskly across informational landscapes whose vastness would otherwise be overwhelming. The lead article in this issue relates descriptions of a few such systems and traces their origins, and the origins of artificial intelligence, to the notion of logic originated by the ancient Greeks.

IR012189

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY
John J. Simon
TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

Information Technology
QUARTERLY

Editor
John J. Simon, Jr.

Information Technology Quarterly
Editorial Committee

John T. Bethell
Editor, *Harvard Magazine*

Madeleine L. Butler
Butler, Raila & Company

Zack J. Deal
Text Processing Specialist, Office for Information Technology

Edward B. Kellogg
Senior Applications Programmer/Analyst

Daniel J. McCarron
Associate University Publisher

Information Technology Quarterly is published seasonally by the Harvard University Office for Information Technology at 1730 Cambridge Street, Cambridge, Massachusetts, 02138, telephone (617) 495-5633. Subscription is twelve dollars per year for four issues; members of the Harvard University community, and educational and non-profit institutions and organizations that make request on their letterhead, are elegible for complimentary subscriptions.

Information Technology Quarterly is prepared in machine-readable form and typeset and printed by the Harvard University Office of the University Publisher.

Material contained in *Information Technology Quarterly* may be copied by non-profit organizations for educational or instructional use provided that the Harvard University copyright notice, the title and date of the issue, and the author, if named, appear on each copy. Copying for any other purpose or for external distribution, and reprinting or republication in another document or periodical, of material contained herein requires written permission from the publisher. Additional copies of this issue, or of back issues, are available from the publisher at three dollars per copy.

Copyright 1985
by the President and Fellows
of Harvard College

ISSN 0885-4246

Letters

Typesize increased

February 1986

With this issue, the type size for most of the body copy for the *Information Technology Quarterly* has been increased one-half point. This modest increase, noticeable only by comparison with the type in earlier issues, was possible without encumbering any significant changes to the established production process and is undertaken in the hope that it will render the publication more legible, particularly for our visually impaired readers.

Editor
Information Technology Quarterly

Newsworthy

Harvard Law School / Digital Equipment Corporation

Artificial intelligence and law

"More is to be gained ultimately by modeling the mind in the domain of law than modeling law using currently known insights about intelligence (artificial or otherwise)."
Michael Dyer

In the fall of 1984, a joint research project was initiated between the Harvard Law School and Digital Equipment Corporation to study the uses of computing technology in the delivery of legal services. The name given to that project, PERICLES, is derived from a ruler of Athens in the fourth century B.C. who introduced many democratic practices including the public hearing of disputes. The letters in the name also serve as a descriptive acronym: Program for Education, Research, and Instruction in Computerized Legal Services.

One of several research areas associated with the project involves exploring the application of artificial intelligence tools and methodologies to the field of housing law and specifically to landlord tenant disputes. We have been actively collaborating with the staffs of the Legal Services Center in Jamaica Plain and the Legal Aid Bureau on campus, both of which are supervised placements for students in the Law School's Clinical Education Program.

The domain of landlord tenant law was chosen as a focus area for a number of reasons:

- housing law represents the largest number of cases facing the clinics and many, due to statutory response time requirements, demand immediate attention;
- the cases can usually be resolved within a relatively short period of time—months rather than years;
- about seventy law students each semester begin working at the Jamaica Plain facility placing high demands for education, supervision, and the answering of necessary yet often repetitive and routine questions on a limited number of staff;
- the expressed desire of clinic directors and staff to be able to serve more clients in an effective and efficient manner;
- the field's interesting combination of statutory laws coupled with decisions from previous cases all of which provide for intellectual challenge;
- and, most importantly, the availability of a number of experienced practioners who are willing to spend time patiently explaining what they do and think about in working with a client case.

These practitioners have been very tolerant of our detailed questioning and methods aimed at explicating the reasoning, knowledge, and skills they use in solving case problems.

Before describing our efforts in more detail, it is appropriate that we pause to momentarily reflect on the efforts of others who have preceded us in looking at the usefulness of artificial intelligence (AI) for the field of law. Although this historical overview will be brief and will not discuss in any depth the technical implementation issues faced by investigators, it will at least provide a context in which to view our work and present readers with a glimpse into the characteristic problems facing those who might contemplate beginning other such endeavors.

A brief history of law and artificial intelligence

In a 1970 issue of the *Stanford Law Review*, Bruce Buchanan, a computer scientist, and Thomas Hendrick, a lawyer, first speculated about the role research in artificial intelligence might play in assisting lawyers both in performing their work tasks and in better understanding

how they reason and solve problems. At that point in time no one had yet begun to apply experiences with artificial intelligence work in other domains to the field of law. Without conducting empirical experiments with lawyers to determine what their reasoning processes involved, the authors outlined what they considered to be four, of several, processes involved in legal problem solving, the end product of which they assumed to be one or more legal arguments:

- establishing conceptual linkages in the pursuit of client goals;
- recognizing and characterizing facts;
- resolving rule conflicts to select which rule applies to the facts at hand;
- finding analogies.

While Buchanan and Hendrick cited artificial intelligence work in other domains that partially addressed these issues, attempts to directly transfer these results and techniques to the legal field pointed up several unique problems. One of the paramount difficulties presented to artificial intelligence research in law is the heavy reliance lawyers place on the comprehension and use of natural language with all its rich semantics. Natural language processing has proven to be a particularly difficult problem for computer science. Coupled to this is the quandary of how best to organize and index information about facts, precedents, rules, statutes, courts, and people in memory so that recognition and characterization of the relevant issues in a new case can occur. Since what is considered to be a fact is heavily context dependent in law, changes in context usually require a re-categorization of the facts along with a re-defining of the conceptual relationships linking previously experienced cases.

Buchanan and Hendrick conclude their paper with an encouragement for interdisciplinary collaboration between lawyers and computer scientists to explore the computer's potential for the legal field. Benefits resulting to both disciplines were foreseen from an explicit understanding of legal reasoning and problem solving. They believed research findings would push on the outer bounds of computer capabilities while simultaneously providing insights for lawyers into their own modes of thinking. The article encourages and challenges other investigators to further elaborate and refine the ideas they outlined.

Somewhat sobering today, after more than fifteen years, is the fact that we still do not have a robust understanding of the complexities of legal reasoning. Nor have computer scientists yet been able to provide the tools and languages required to deal with the issues Buchanan and Hendrick cite above except for problem domains that are fairly well structured. This is not to say, however, that much has not been learned and accomplished in furthering the twin goals of understanding legal reasoning and extending artificial intelligence capabilities. It is clearly the case that, despite the hype associated with artificial intelligence today, a great deal of interdisciplinary work remains not just with lawyers and computer scientists but with philosophers, linguists, anthropologists, cognitive scientists, and neuroscientists as well.

During the past decade and a half only a relatively small number of individuals have taken up the challenge of Buchanan and Hendrick in applying artificial intelligence to law and most of these have worked in rather circumscribed ways attacking pieces of the set of problems. In 1972 Peter B. Maggs and Cary G. deBessonet focused on formalizing language so that specific questions about legal rules as they pertain to liability law could be answered by a computer, and, in 1975, W. G. Popp and B. Schlink developed a rule base system based on the German Civil Code for negligence law. As Anne Gardner points out, both of these early efforts lacked distinct components in which the facts of the cases could be represented separately from and then applied to the legal rules or statutes.

The first attempt to represent multiple types of legal knowledge was done by Jeffrey Meldman during his doctoral thesis work at the Massachusetts Institute of Technology in 1975. Meldman, who earlier had received a JD from the Harvard Law School, focused on the precedent law of assault and battery cases. The problem characteristics of the domain forced consideration, and thus representation, of important precedent cases, relevant legal rules, and their application to the imputed hypothetical case facts. Meldman's program built up rudimentary models of legally relevant human activities and laws which were hierarchically classified into categories. A matching process within these categories resulted in the case being analyzed, either by syllogism or by proximity with a simplified analogy mechanism, on the basis of its hierarchical position

In 1970, Bruce Buchanan and Thomas Hendrick first speculated about the role research in artificial intelligence might play in assisting lawyers in performing their work and in better understanding how they reason and solve problems.

in relation to the relevant rules.

Thorne McCarty also utilized separate representations for the facts and applicable laws from the field of corporate tax law for reorganizations in his Taxman I system. A model was constructed of the facts of tax cases and the relevant concepts of the Internal Revenue Service code. The output from the computer was an analysis of whether or not a reorganization had in fact taken place.

McCarty observed several limitations in his system. Taxman I could not handle any "open textured" concepts or model the creation and evolutionary changes that concepts undergo through use over time. These realizations led to the development, with N. S. Sridharan, of Taxman II, which sought to remedy these deficiencies. Using a frame based language, McCarty and Sridharan developed a system of template prototypes of the objects in the domain along with relationship links between the objects. Facts were seen as instances of these prototypes in a specific context. Descriptions and productions, arranged hierarchically, were used to represent the applicable rules and concepts. The "amorphous" or "open textured" concepts created and modified in the judicial process were represented by what the researchers called "prototype-plus-deformation" structures. These structures contained the template with its necessary conceptual conditions as series of concept examples, and a series of operators for modifying these examples.

Another project initiated in the late 1970s was Donald Waterman and Mark Peterson's Legal Decisionmaking System (LDS). These researchers utilized the expert system model of a rule based production system program comprising a series of antecedent-consequent, or if-then, rules. The rules contained the knowledge and skills embodied in the decisions of claims adjusters and lawyers involved in working out settlement values for product liability negotiation cases. Waterman and Peterson were interested in studying the relationships among five categories of rules they created as these were used by decision makers, and are continuing their work today as it applies to the settlement values of claims involving asbestos injuries.

Carole Hafner's thesis project in 1978, the Legal Research System (LRS), focused on the development of a knowledge based document retrieval system. Using a semantic network representation, texts relevant to a particular legal problem involving negotiable instruments could be retrieved. This was one of the first attempts to focus on the problems of representing conceptual knowledge for retrieval, a task considerably more difficult than that of providing the more limited ability of key word retrieval systems such as WESTLAW and LEXIS.

Others have made efforts along these lines as well, most notably deBessonet and George R. Cross at Louisiana State University. Their CCLIPS (Civil Code Legal Information Processing System) project is aimed at modeling part of the contract law of the Civil Code of Louisiana. DeBessonet and Cross have developed a language for representing legal rules and concepts called ANF, for Atomically Normalized Form, that also handles some causal inference to aid in the conceptual retrieval of legal information. Rather than understanding or parsing natural language inputs, CCLIPS reads in descriptions of facts written in ANF and tries to conclude what legal consequences follow from the rules. ANF works by manipulating the explicit surface level information in such a way that deeper representational schemes can be generated. Realizing that they cannot expect lawyers to learn ANF, deBessonet and Cross are currently working to develop an interface that will parse a limited subset of natural language inputs into ANF.

By the beginning of the 1980s, a consensus of experiences involving artificial intelligence and law had accumulated to the point of being articulable in the following somewhat heuristic statement by Sandra Cook et al:

> In one way or another, a legal consultation system must represent the "facts" of a case at a comfortable level of abstraction, and it must represent the "law" in the chosen area of application, where the law consists of a system of "rules" and "concepts" which specify the rights and obligations of the parties in the case. Legal analysis, in its simplest form, is then a process of applying the "law" to the "facts".

One should not be lulled into thinking that this heuristic recipe for conceptualizing the legal process addresses the very difficult issues of how to represent either the law or the facts, especially when legal case facts "involve all the complexities of daily life: human actions, beliefs, intentions, motivations, etc., in a world of ordinary objects like houses and automobiles, and complex institutions like businesses and courts" (Cook et al). A serious problem is defining what the "law" is. Is

In 1972, Peter Maggs and Cary deBessonet focused on formalizing language so that specific questions about legal rules as they pertain to liability law could be answered by a computer.

it an amalgam of statutes? Case law? Opinions or politics? Some combination of these? Is legal reasoning an attempt to logically deduce consistent rules from this amalgam?

Cook et al go on to discuss several "strategic choices" that designers of an artificial intelligence system for law must face. One decision has to do with the breadth and depth of the legal subject area under study. Usually the broader the range a system is capable of responding to the shallower its knowledge will be. Conversely the narrower the system's area of legal range the deeper, usually, the coverage one can expect to see in the system. Related to this issue is the problem of deciding whether to use a representational language that is relatively simple syntactically or semantically quite rich. This has very strong consequences for how to represent legal cases in a knowledge base. These questions require designers to define a legal conceptual model that allows for considerations for building a knowledge acquistion facility to help input the data as well as functionality for modifying the legal conceptual model as laws change or new features gain prominence in cases over time. These issues constitute a summary of the major problematic obstacles and opportunities that faced researchers in artificial intelligence and law as the 1980s began.

There are several more recent efforts that, because of their ambitious scope, deserve close attention. One of these projects, also, grew out of a thesis effort, this time at Stanford. Gardner started with the goal of bringing into one system the ability of deductively making decisions from rules and reasoning about case comparisons analogically. She sees a legal reasoning program as thus having two phases: "first, heuristically settling the obvious points and identifying the serious issues, the 'hard questions' in the case; and second, finding arguments on both sides of the hard questions." She contends that reasoning analogously to previous cases should only occur when these so-called "hard questions" arise. She chose to study the case law domain of contracts by offer and acceptance using cases from first year law school exams. Since offer and acceptance problems use dialogue, Gardner also had to confront the issue of developing a facility for handling natural language input (Gardner 1983). As she discovered, this ability was necessary as well in handling the "open textured" nature of many legal concepts.

Another significant effort to apply artificial intelligence to the field of law, underway at the University of Massachusetts at Amherst, involves Professors Edwina Rissland, Wendy Lehnert, and David McDonald. The system, called COUNSELOR, has several integrated capabilities for natural language understanding, generation, and discourse. The legal component, called HYPO, focuses on modeling legal reasoning using example-based or case-based techniques. The legal domains studied are software trade secrets and, to a limited extent, contract law. Given the fact situation presented by a client, the goal of the system is to develop arguments a lawyer might construct along with the facts and previous cases that can be used to support them. More recent efforts have focused on having the system begin to identify arguments which the opposing counsel might make and to suggest initial tactics for weakening them (Ashley and Rissland). The University of Massachusetts team has taken the tack of building functionality into the sytem through the construction of separate modules that, thus far, all utilize the same knowledge base and frame based representational language techniques.

Engaged in a related effort at the Georgia Institute of Technology are Janet Kolodner and a group of graduate students whose MEDIATOR system incorporates a process model for the types of case based reasoning used in interpreting and resolving common sense disputes. Their research highlights the benefits of an artificial intelligence system that is able to: consider, through the use of analogical reasoning, previous experiences as a way to learn from errors; reduce the number of steps to complete a task; aid in learning new abilities; make predictions; construct explanations. A critical component of the ability to utilize previous experiences involves the problem solver interacting with a memory for experience. This episodic memory must not only be indexed in a useful way but also be dynamic and subject to modification and have the ability to be "reminded" of experiences previously encountered (Kolodner) (Schank). In addition, the memory system must have the ability to differentiate a new experience from similar ones, to focus the problem solver's attention by analogously generating expections of features to look for, and to compile similar experiences (episodes) into a new memory structure. MEDIATOR uses a frame based language to represent

DeBessonet and Cross have developed a language for representing legal rules and concepts called ANF, for Atomically Normalized Form, that also handles some causal inference to aid in the conceptual retrieval of legal information.

sets of schemata that integrate the functionality of problem understanding, problem solving, and memory.

The Artificial Intelligence Lab of the University of California at Los Angeles (UCLA) has recently established the Foundations of Legal Expertise research project, which is concerned with "the design and implementation of computer programs capable of modeling various aspects of the cognitive processes of legal experts" (Dyer and Flowers). The group has outlined a broad and long term research agenda that encompasses the following:

> From a cognitive modeling point of view, law offers a wealth of areas to explore, including:
>
> - natural language comprehension and generation
> - organization of episodic memory
> - indexing, accessing, and retrieval of episodes from memory
> - memory modification and learning through experience
> - planning, counter-planning, and strategy formation
> - argumentation, belief maintenance, and persuasion
> - reasoning, common sense, and the formation of judgements
> - processes of questioning, examination, and summary formation.

Michael Dyer and Margot Flowers' approach to the study of law and artificial intelligence relies heavily on work they and others, (e.g. Kolodner's work on memory and Lehnert's work on natural language understanding), performed previously as Ph.D. candidates in Roger Schank's artificial intelligence laboratory at Yale.

Many of the individuals cited above—Rissland, Kevin Ashley, Dyer, Gardner, McCarty, and Waterman—participated in a panel discussion on "AI and Legal Reasoning" held last summer at the International Joint Conference on Artificial Intelligence (IJCAI). The following comments represent a sampling of the statements made by the panelists as they reflected upon the characteristics of the American legal system and the problems they pose for present day artificial intelligence research:

> It's more of a "scruffy" domain than a "neat" one, despite its orderly rule-like surface veneer.
>
> Complex conceptualizations are expressed in natural language.
>
> Giving legal advice often starts with hearing a "story" where the client was one of the actors.
>
> Planning in this domain is complex.
>
> Anglo-American jurisprudence . . . makes use of a large body of concrete EPISODES. Becoming a lawyer requires learning how to index, generalize, and apply these concrete episodes in service of reasoning and argumentation strategies.
>
> Law is simply an attempt to formalize COMMONSENSE reasoning and notions of justice, morality, and fair play.
>
> Law is a domain which is largely case or example-based and thus examples are truly central.
>
> Law . . . has the unique property of being pseudo-formalized . . . rules are often deliberately ambiguous, contradictory, and incomplete. Many of the actual rules used in legal reasoning are rules about how to access and reason with the "formal" rules.
>
> Law is in a constant state of change.
>
> Law . . . is unusual in that we have to expect the experts to disagree.
>
> Legal rules are written in natural language, and the problem of saying how they apply to a particular situation is in large part a problem of saying what this language means.
>
> The situations to which we might want to apply legal knowledge encompass practically the whole range of human activity—with a special focus on those human situations where something goes wrong.
>
> The law's handling of open textured predicates is not neutral, that is, the decision about the concept is tied in with who the decider wants to win the case.
>
> There are several uses of analogy.
>
> Much of legal reasoning . . . is a memory indexing and search problem rather than a logical or "theorem-proving" problem.
>
> Process models of legal expertise must be designed as a whole where beliefs, goals, planning, and arguing are integrated.
>
> The most critical task in the development of an intelligent legal information system, either for document retrieval or for expert advice, is the construction of a conceptual model of the relevant legal domain.
>
> AI and law require further progress in fundamental AI and Cognitive Science research on issues like representation.
>
> Successful work in law and AI—both theory and application—requires much better "conceptual models" of the legal domain than the field of AI can currently provide "off-the-shelf."

As one considers these remarks in light of efforts over the past fifteen years, one cannot but be struck that, despite the complexity of the projects going on today and the lessons learned, what has emerged most vividly is the sense that it has taken us this long to have merely sketched out the terrain of problems requiring attention in the domain of law.

A University of Massachusetts team consisting of Professors Edwina Rissland, Wendy Lehnert, and David McDonald has taken the tack of building functionality into a sytem through the construction of seperate modules that all utilize the same knowledge base and frame based representational language techniques.

What was initially attacked by pairs of researchers now occupies the time and thought of several individuals, usually with a number of additional graduate students, from a variety of disciplines working in long term collaboration.

Research into the cognitive processes of legal reasoning

Despite the many references cited above that have called for a more robust understanding of the cognitive processes of lawyers, there is a surprising sparseness to the literature on the subject. Not only are there few empirical investigations of legal reasoning upon which to draw, but what lawyers call "legal reasoning" usually involves observations of judicial decision making. Even the research efforts mentioned above are focused primarily on observations of the task environment lawyers work in, that is, on what they do. Detailed studies of the reasoning and problem solving processes of lawyers that support the assumptions made in building the artificial intelligence systems referred to are lacking. A close reading of these artificial intelligence and law projects clearly highlights a number of cognitive processes, treated today as "black boxes," that can be informed by sound empirical studies.

Only two studies focusing on the cognitive processes of lawyers have been widely reported. Michael G. Johnson and Paul E. Johnson at the University of Tennessee are focusing on explicating the processes involved in lawyer competence. They are working to shed light on the problems involved in recognizing and certifying lawyers for qualification in certain specialized areas of law. Their approach grew out of the realization that despite rational and logical arguments, debates, and conferences, the question of what competence means in a specialized field cannot be answered without the support of detailed empirically based data. We believe this to hold, too, for attempts at using artificial intelligence in the legal domain. As long as the emphasis remains primarily on observing what lawyers do in performing tasks we only have a partial understanding of what is occurring. As Gardner states, "most [of the projects in AI and law] are concerned with what moves are legitimate in legal reasoning, not with how any particular individual comes to find them."

The second project with a cognitive focus is that of J. F. Stratman who investigated the reasoning utilized by lawyers in the construction of legal arguments to be used in court.

Certainly the research of Kolodner on episodic memory organization and analogical reasoning, of Rissland and Ashley on examples and analogies, of Dyer and Flowers on the cognitive processes of legal experts, and the fruits of Schank's artificial intelligence laboratory have contributed much to the enrichment of our understanding of cognition and cognitive processes as they pertain to the field of law, but more studies are needed.

Law and artificial intelligence at Harvard Law School

At the Harvard Law School, we are committed to working and studying in the area of law and artificial intelligence over the long term. Our focus is interdisciplinary in nature; collectively our academic and work experience spans the disciplines of law, psychology, philosophy, and artificial intelligence. We believe these multiple perspectives add richness and depth to considerations of the problems and unique characteristics of law. We each take a different yet complementary tack on the project. The approach followed by George Wood can be termed the engineering model, that of Peter O'Neil the cognitive science model. Both orientations emphasize what has come to be known as "knowledge engineering." We will outline our engineering approach first.

As previously mentioned, the goal of artificial intelligence research in the context of Project PERICLES is to study the usefulness of the application of artificial intelligence techniques to the legal domain of landlord tenant law. Since "expert systems" are the best understood and most practical form of current artificial intelligence technologies, our specific research focuses on the development of an expert system. That is, rather than attempting to build a system capable of modeling all of legal reasoning in a general problem solving manner, we are focusing on legal expertise in the very specific domain of housing law.

Issues for research and development include: the exploration of knowledge representation strategies for assessing appropriate methods for representing legal expertise; the design of appropriate explanation and automatic knowledge acquisition facilities to allow the system to acquire additional legal expertise easily through direct interaction with an expert; and the potential

Janet Kolodner and a group of graduate students at the Georgia Institute of Technology are developing a process model for the types of case based reasoning used in interpreting and resolving common sense disputes.

uses of such systems as instructional tools.

Expert systems are appropriate when there is (1) an established body of expertise, (2) a shortage of experts, and (3) a set of tasks that experts are repeatedly called upon to perform. In the clinical setting there is a shortage of expert lawyers; in order to capitalize on their expertise, legal novices (interns, students, and paralegals) are given case specific tasks which require expert attention in the form of coaching or review. Thus, much of the expert's time is spent reviewing and directing the work of novices. Because there is an established body of expertise in the areas of housing law and landlord tenant law and a large portion of the resources of the clinics is devoted to housing law cases, it seemed appropriate to build an expert system to advise in housing law case preparation.

The greatest problem for expert systems development has traditionally been the amount of effort required to add new knowledge to a system. The study of automatic knowledge acquistion is thus one of the most exciting and fruitful research areas in artificial intelligence. Several knowledge acquisition strategies seem viable in the domain of legal practice, including learning by being told and using feedback to refine or correct existing knowledge. Knowledge acquisition will allow expert lawyers to teach the system without having to become programmers or knowledge engineers.

The presently proposed system, NOMOS from the ancient Greek concept for law, is to serve as an intermediary between experts and novices in preparing defenses in housing law. The system is designed to acquire knowledge from experts and use this knowledge in assisting novices.

Experience in knowledge engineering has shown that the rapid development of a number of preliminary prototypes, each incorporating a small segment of domain knowledge, resolves major issues in knowledge representation and shapes specific features of system architecture. These prototypes lead to the development of a "kernel" system, capable of handling a constrained range of cases, with designed-in expandability.

The first prototypes focus on straightforward cases: a paradigmatic example of a case is examined and a "tiny" system capable of handling it is designed, then extended gradually to handle variations on that theme. Thus prototypes are said to be "deep" since they do not handle a broad range of cases. A measure of a prototype's success is the degree to which it can be extended to handle new cases.

Prototypes are developed primarily for the purpose of exploring knowledge representation and knowledge acquisition. We have completed several prototypes which demonstrate central features and weaknesses of several knowledge representation schemes and knowledge acquisition techniques.

Description of the NOMOS system

NOMOS will be the first full-scale system to integrate successful features from the prototype systems and serve as a base system for future research and development, as well as provide a stable environment for building a usable system, that is, one with broad capabilities rather than just a few case-types. This section is to provide a description of the desired performance of the system from the point of view of potential users. The focus here is on what is to be done by the system rather than on how it is to be done. The section that follows discusses implementation methods, modules, and system integration.

The novice is expected to prepare as much of a case as possible, partly for the experience of doing so and partly because of the relative expense of direct expert involvement. This task has several components: client interviewing and information gathering; recognizing possible defenses and counter-claims; and document assembly. Expert attention is most often required in several aspects of recognition, that is, in recognizing possible actions and knowing what information is required to determine whether they are appropriate in a given context. The range of these actions includes the construction of arguments, rejoinders, defenses, claims and counter-claims, to "simply" suggesting lines of negotiation.

Beyond the obviously computerizable tasks of record keeping, data management, and document production, an intelligent system should be capable of serving as an intermediary between experts and novices, of "coaching" the novices. Specifically, the system should help the novice in recognizing defenses by applying expertise. Further, the system should be able to learn about new situations through direct interaction with an expert.

In the role of intermediary, the

Michael G. Johnson and Paul E. Johnson at the University of Tennessee are working to shed light on the problems involved in recognizing and certifying lawyers for qualification in certain specialized areas of law.

system is primarily described as a medium for information transfer between experts and novices, with subsidiary roles in record keeping and document production. The transferred information includes knowledge of facts, rules, and procedures, and the mediation is characterized by a dialogue of information-gathering queries and offered advice, and explanations of both. In order to flesh out this description, we must consider system functions from three perspectives: the novice interface; the expert interface; and subsidiary functions.

The novice user interface to NOMOS

A novice should view NOMOS as an assistant and tutor capable of keeping client records and providing advice on the treatment of a case. This role should be viewed as part of a "dynamic" system in an ongoing process rather than as a "one-shot" activity. Because of the need to be able to deal with cases in stages, and the desire to avoid entering the same information on several occasions, the record-keeping and advisory functions should be strongly integrated, at least from the user's viewpoint.

We assume that the novice user has some legal knowledge and familiarity with the system but is not a domain expert; typical novice users would be paralegals, legal interns, and student assistants. A reference manual and tutorial should be available, along with on-line help and a brief course for first-time users. Novices should not require extensive training to use the system.

NOMOS is intended to provide a novice with advice about how to handle a case. Ideally it will offer a summary of a defense strategy together with a moderately detailed plan of action, and suggest hypothetical subsequent actions depending on contingencies. It will save what it knows about a case so that as more information becomes available (for example, new evidence, results of proceedings, etc.) it can provide more advice, and so that results of advice can be applied to improved performance. NOMOS will be capable of responding to a user's queries about, for instance, the advice it offers and the questions it asks. Initially, this will be restricted to "why-questions" asked immediately following a system action.

The system should offer a tutorial mode for teaching by reviewing sample cases. Tutorials should provide substantive educational benefits, not just teach students how to use the system itself. In tutorial mode, the system should automatically offer or provide explanations of each issue encountered.

In a typical session, a novice will interview a client using NOMOS as a "smart" notebook. The system will maintain a display of keywords indicating missing information, and the user will be able to enter this information or ask for a description of the information required, an explanation of why the information is needed, or a menu of possible answers. The system should offer explanations of its actions, queries, and advice by backtracking to its reasons for them, by showing relevant citations, and by displaying specific explanatory text entered by the expert.

The expert user interface to NOMOS

An expert should be able to use the system for reviewing cases entered by novices, and for adding, refining, and correcting system knowledge. The system should offer several forms of knowledge acquisition, including "procedural knowledge acquisition" and "analytical knowledge acquisition." These techniques should be usable at different stages, from initial entry of a new case type, where they contribute to "learning by example," to the fine-tuning stages of "detail revision." In its knowledge acquisition mode, the system should behave like a curious student.

Experts with sufficient experience with NOMOS may learn to use the analytical programming tools described in the next section, though these tools will initially be designed for use by programmers rather than by domain experts.

Subsidiary functions

Other functions of the system which will contribute to it's overall usefulness center on database and output features and on tools for evaluating and modifying features of the system itself.

Record Keeping—The system should maintain a database of cases handled, including information required to allow experts to review cases in detail, and allow case data to be updated as progress is made. The database should also provide the ability to perform statistical analyses of features of cases, and record comments by novices and experts for use as guides to revisions. Initially, this database will be kept in ASCII format for ease of development. This will enable users to perform statistical analyses using existing packages such as MINITAB or SPSS.

Document production—NOMOS should provide appropriate docu-

NOMOS, from the ancient Greek concept for law, is to serve as an intermediary between experts and novices in preparing defenses in housing law.

ments for the actions it advises its users to perform. We assume a text database with provision for variable substitution and conditional text inclusion will be available, and that NOMOS will provide output to generate documents by providing appropriate input to this text production system. If necessary, such a system could be developed independently, but since there are existing systems with this capacity it seems preferable to create an interface to one (or more) of them.

Analytical programming tools—Tools will be provided to enable system programmers with knowledge of the internals of the system to debug it and trace both normal and aberrant behavior in it. These tools should provide displays of both system and knowledge-base logic, and should especially allow the examination of related propositions for gaps, overlaps, and internal conflicts, even though the knowledge acquistion device is supposed to avoid these. In the future, some of these tools may be refined to the point that domain experts, without system-programming abilities, will be able to use them.

Description of cognition and law research

Our cognitive science approach has a two-fold orientation: first, to explicate the reasoning processes and types of knowledge that an expert practioner utilizes in problem solving in landlord tenant cases; and second, to understand what law students learn and remember about their three years in school. Do they develop and refine conceptual models about law and how the American legal system operates? Do they remember large numbers of specific cases? General categories or specific rules? Stated differently, what kinds of legal information do students begin to use as organizing concepts in memory? Do they build up disparate information "pools" across the subfields of law or does information accumulate in some "unified" way? If separate "pools" or "islands" of knowledge exist, when, if at all, are links formed between them? Does some type of informational "restructuring" occur as students' knowledge of law expands and becomes refined over time?

Through the use of protocols, the thinking aloud recordings of problem solving behavior (Simon and Ericsson), and interviews conducted over extended periods of time with two experienced attorneys in landlord tenant law, we have amassed a rich accumulation of data about expert cognitive processes. This data is currently being analyzed to develop ideas about the fundamental cognitive processes involved in expert legal reasoning.

The study of what law students actually learn and organize in memory, what comprises their legal knowledge and skills, is obviously a long term endeavor. Initial data will be gathered from volunteer students from each of the three years of classes who will be asked to comment about those classroom experiences which have become "memorable" for them in developing ideas and notions about what law means. The design of this research has begun and we expect to begin gathering student responses within a month.

Several research issues of special interest are interwoven through both of these cognitive science approaches. These include knowledge and mental representation; the use of images with conceptual models; episodic memory organization; the indexing and accessing of, and retrieval from, memory; and memory restructuring as a result of experience.

In our approach, the "engineering" research issues and the "cognitive science" research issues overlap considerably with each helping to inform the other. Our research values both top down analysis and bottom up results. This is purposeful and we believe will help to move our understanding of law and legal reasoning further along than each of us could working alone. As Derek Bok, President of Harvard University, remarked last spring in his report to the Board of Overseers:

> Over the past twenty years, computers have proved to be a major stimulus for eliciting work and thought about teaching methods and the processes by which human beings learn. . . . A striking fact about higher education is the lack of systematic effort to understand the ways in which students learn. Faculties rarely spend time thinking collectively about the educational process. When they do, they almost always discuss *what* students should read and study rather than *how* students learn and how they might be helped to learn more effectively.

We believe our work in PERICLES to be a step towards a more informed discussion of *what* students learn about law and *how* they reason about it.

The goal of artificial intelligence research in the context of Project PERICLES is to study the usefulness of the application of artificial intelligence techniques to the legal domain of landlord tenant law.

The authors

Peter O'Neil is a principal engineer in the External Research Program, Corporate Research & Architecture, of Digital Equipment Corporation. His current assignment involves research at the Harvard Law School under Project Pericles. His particular interests include developing models of lawyer reasoning and the application of artificial intelligence to law.

George Wood is a Senior Research Associate at the Harvard Law School. Prior to coming to Harvard last September, Dr. Wood was a Senior Research Programmer at Carnegie-Mellon University. His principal interests are knowledge acquisition in expert systems, philosophy of psychology, and epistemology. He is currently involved in phase III of Project Pericles, an expert system for housing law.

Bibliography

Artificial intelligence and law

Ashley, K., and E. Rissland. "Towards Modelling Legal Argument." In *Italy Proceedings*, COUNSELOR Memo, (1985).

Ashley, K., E. Rissland, and E. Valcarce. "Explaining and Arguing with Examples." In *Proceedings, American Association Artificial Intelligence* (1984).

Bok, D. *The President's Report to the Board of Overseers of Harvard College*. Cambridge, Massachusetts: Harvard University, (1985).

Buchanan, B., and T. Headrick. "Some Speculation about Artificial Intelligence and Legal Reasoning." *Stanford Law Review* 23 (1970): 40-62.

Cook, S., C. Hafner, L. McCarty, J. Meldman, M. Peterson, J. Sprowl, N. Sridharan, and D. Waterman. "The Applications of Artificial Intelligence to Law: A Survey of Six Current Projects." In *Proceedings, AFIPS National Computer Conference* (1981): 689-696.

deBessonet, C., and G. Cross. "An Artificial Intelligence Application in the Law: CCLIPS, A Computer Program that Processes Legal Information." Computer Science Department Report 85-003, Louisiana State University, (1985).

deBessonet, C., and G. Cross. "Representation of Some Aspects of Legal Causality." Computer Science Department Report 84-006, Louisiana State University, (1984).

Dyer, M., and M. Flowers. "Toward Automating Legal Expertise." *Computing Power e) Legal Reasoning.*

Edited by C. Walter. West Publishing Company, (1985): 49-68.

Gardner, A. "Law Applications." Submitted to *Encyclopedia of Artificial Intelligence*. Edited by Stewart Sharpiro. New York: John Wiley & Sons, (1985).

Gardner, A. "An Artificial Intelligence Approach to Legal Reasoning." Department of Computer Science Report STAN-CS-85-1045, Stanford University, (1984).

Gardner, A. "The Design of a Legal Analysis Program." In *Proceedings, American Association Artificial Intelligence* (1983): 114-118.

Hafner, C. "An Information Retrieval System Based on a Computer Model of Legal Knowledge." Ph.D. thesis. University of Michigan, Ann Arbor, 1978.

Johnson, M., and P. Johnson. "Competence and Specialization in the Practice of Law: An Empirical Investigation of Lawyer Reasoning." Project Summary, National Science Foundation, 1981.

Kolodner, J. "Maintaining Organization in a Dynamic Long-Term Memory." *Cognitive Science* 7, (1983): 243-280.

Kolodner, J., R. Simpson, and K. Sycara-Cyranski. "A Process Model of Case Based Reasoning in Problem Solving." In *Proceedings, International Joint Conference on Artificial Intelligence*. University of California at Los Angeles, (1985): 284-290.

Lehnert, W. *The Process of Question Answering*. Hillsdale, New Jersey: Lawrence Erlbaum Associates, (1978).

Maggs, P., and C. deBessonet. "Automated Logical Analysis of Systems of Legal Rules." *Jurimetrics Journal* 12, (1972): 158-169.

McCarty, L. "Reflections on TAXMAN: An Experiment in Artificial Intelligence and Legal Reasoning." *Harvard Law Review* 90, (1977): 837-893.

McCarty, L., and N. Sridharan. "The Representation of an Evolving System of Legal Concepts: II. Prototypes and Deformations." In *Proceedings, Seventh International Joint Conference on Artificial Intelligence*. Vancouver, Canada, (1981): 246-253.

McCarty, L., and N. Sridharan. "The Representation of an Evolving System of Legal Concepts: I. Logical Templates." In *Proceedings, Third Biennial Conference of the Canadian Society for Computational Studies of Intelligence*. Victoria, British Columbia, (1980): 304-311.

Meldman, J. "A Structural Model for Computer-Aided Legal Analysis." *Rutgers Journal of Computers and the Law* 6, (1977): 27-71.

Popp, W., and B. Schlink. "JUDITH, a Computer Program to Advise Lawyers in Reasoning a Case." *Jurimetrics Journal* 15, (1975): 303-314.

Rissland, E. "AI and Legal Reasoning." Report of a panel. In *Proceedings, International Joint Conference on Artificial Intelligence*. (1985).

Schank, R. *Dynamic Memory*. Cambridge, England: Cambridge University Press, (1982).

Simon, H., and K. Ericsson. *Protocol Analysis: Verbal Reports as Data*. Cambridge, Massachusetts: MIT Press, (1984).

Stratman, J. "Studying the Appellate Brief and Opinion Composing Process: A Window on Legal Thinking." *Juris* 19, no. 1 (1984): 9-14, and no. 2 (1984): 12-19.

Waterman, D., and M. Peterson. "Models of Legal Decisionmaking." Institute for Civil Justice Report R-2717-ICJ, Rand Corporation, 1981.

Artificial intelligence: a long and winding road

Many hundreds of years elapsed between the debut of talking oracles and dancing mannekins and the development of the prototypes of the analog and digital computers; some two hundred years of experimentation with calculating machines preceded conception of the general-purpose computer, another one hundred years elapsed before the concept was resurrected, and yet a few more years preceded its realization; meanwhile, formalization of an algebra of thought lagged some two hundred years behind the origin of the notion and nearly another century passed before such an algebra was applied to the switching of electrical circuits such as are found in computers; and all of this is whence artificial intelligence derives.

Illustrations by Eleanor Bradshaw/Office of the University Publisher

"Be just what you iss, not what you iss not; 'cause what you iss, iss da happiest lot." This advice Mr. Wizard, a character in a circa 1960s Saturday morning cartoon program called the King Leonardo Show, proffered repeatedly to fellow character, Mr. Turtle, each time he returned the latter from another disastrous adventure in the role of someone else. The field of artificial intelligence, given the debate and occasional rancor that has intermittently characterized it over its brief twentieth century history, might be stood in good stead by heeding such advice; that is, the happiest lot, as far as artificial intelligence is concerned, might very well be to be just what is, and to decline any pretence of being what it is not — yet.

The qualification is warranted. Like computing, the field of artificial intelligence, in the modern sense of the term, is young; it is only now entering the hands, and heads, of a third generation of students, and many of the first generation still linger. Computing itself has been both sustenance and sedative for artificial intelligence; computing machinery and techniques are the "stuff" in which "artificial intelligence" would be realized, yet computer manufacturers were, for a time, among the most adamant decriers of the notion that machines might "think". Though born together, computing and artificial intelligence are at widely disparate stages of development; the latter is only now trundling a pushcart full of "expert systems" into a marketplace teeming with retail and mail order purveyors of computing's extraordinarily scaled up capabilities and dramatically scaled down packaging.

This is not to imply that things should be other than they are. Indeed, as it has turned out, realization of much that research in artificial intelligence has sought to accomplish has been contingent on the evolution of more appropriate and more powerful machinery. Nor is it suggested that artificial intelligence should have heeded Mr. Wizard's

advice sooner. The reasoned debate and argumentation alluded to above will very likely, in retrospect, be deemed to have contributed to the enrichment of our understanding of human thought as much as, and possibly more than, actual machine-centered research carried out in the same time frame.

Though artificial intelligence as a field or discipline is relatively young, only now entering the hands and heads of a third generation of students, the notion of, and debate over, mechanization of human thought is many centuries old.

Though artificial intelligence, as a field or discipline, is relatively young, the notion of, and debate over, mechanization of human thought is many centuries old. Several contemporary authors with diverging views on artificial intelligence explore to varying degrees its antecedent myth and reality, among them, Stanley L. Jaki and Hubert L. Dreyfus, who write with a wariness of the terminology and its implications, and Frank Rose and Pamela McCorduck, whose prose strokes it with sensitivity, if not affection.

By some accounts, the first to entertain the notion of something akin to artificial intelligence lived several centuries B.C. McCorduck (1979, 4-5) recounts briefly, from Greek mythology, the story of crippled Hephaestus—son of Hera, god of fire, and divine smith—who variously fashions a bronze gentleman to patrol the beaches of Crete, a set of wheeled tripods capable of propelling themselves to and from Olympian feasts, and, to help him walk and assist him in his forge, a host of attendants for whom she turns to Richard Lattimore for a description:

> These are golden, and in appearance like living young women. There is intelligence in their hearts, and there is speech in them, and strength, and from the immortal gods they have learned how to do things. (Lattimore 1963, 386)

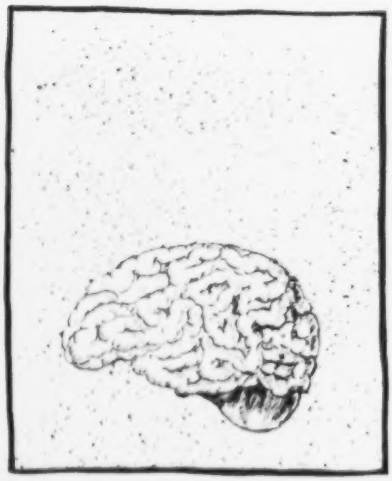
Even before we pose the question of whence intelligence derives, of what is the agent that imbues intelligence to a non-biological entity, we have at least one answer.

The mythology of the *The Iliad* is nearly three thousand years old; the line of reasoning that gave rise to the notion of logic is some four hundred years younger. Dreyfus observes that Plato observed that Socrates was the first to see in the Greek invention of logic a vision of reasoning reduced to some kind of calculation to which all arguments might be submitted for resolution. Dreyfus' rendering of Socrates vision in the terminology of computer science is: "a set of rules which tells us, from moment to moment, exactly how to behave" (1979, 67).

Plato sought to generalize Socrates' vision by suggesting that "all knowledge must be stateable in explicit definitions which anyone could apply" (ibid.), a condition, Dreyfus notes, that leaves out such areas of human thought as "require skill, intuition, or a sense of tradition" (ibid., 68). Aristotle's observation that the very application of the Platonic rules relied on intuition suggested that the system was unamenable to complete formalization. "For the Platonic project to reach fulfillment," Dreyfus remarks, "one breakthrough is required: all appeal to intuition and judgment must be eliminated" (ibid.).

The machinery of artificial intelligence to the fifteenth century consisted largely in statues variously imbued with such attributes as movement and speech.

Jaki posits as the firmer footing attained a millenium and a half later by the mechanistic philosophy that began its precipitous climb from Plato's line of reasoning the realization that "for physics to make progress and sense, it must exclude from its range anything non-quantitative" (1969, 20). "As Galileo discovered that one could find a pure formalism for describing physical motion by ignoring secondary qualities and teleological considerations," Dreyfus explains, "so, one might suppose, a Galileo of human behavior might succeed in reducing all semantic considerations (appeal to meanings) to the techniques of syntactic (formal) manipulation" (1979, 68-69).



The real machinery of artificial intelligence to the fifteenth century, as distinguished from the mythical creations of Hephaestus, consisted largely in statues imbued variously with such attributes as movement and speech. These, McCorduck notes, were all as one to the people of the time, who detected "little difference between a human figure that nodded, bowed, marched, or struck a gong at a precise and predictable moment, and a human figure that answered knotty questions and foretold the future" (1979, 10-11).

What ensued, described by McCorduck as "a population explosion of automata" (ibid., 13), yielded within a century the prototypes of the analog and digital computers. "In 1621," Jaki explains,

> William Oughtred placed side by side two logarithmic scales, and by sliding one along the other . . . could perform multiplications and divisions. As there are, in theory at least, infinite numbers of points between any two positions on such scales, the slide rule, as the device

The slide rule, an early seventeenth century invention of William Oughtred, can be considered a continuous function analog computer, and the adding machine of Blaise Pascal its primitive digital counterpart.

> came to be called later, can in effect be considered as a continuous function analog computer. Its primitive digital counterpart, the adding machine, was the product of the inventiveness of [Blaise] Pascal. . . . Though it did not operate smoothly, due to unsatisfactory workmanship, it fully justified the proud remark of its inventor: "A calculating machine achieves results that come nearer to thought than anything done by an animal. But it does nothing to enable us to say that it has will, as we say animals have." (1969, 22-23)

It was in this milieu, of talking statues and rudimentary computers, that René Descartes formulated his mechanistic world view which sought to "devitalize" biological organisms by describing them in terms of complex machines. In treating the human organism, however, although he carried the mechanical analogy all the way to the brain, likening it to an hypothetical hydraulic machine, there he stopped, admitting a dichotomy between bodily organs, including the brain, and "mind." To make his position altogether clear, Jaki observes, Descartes

> referred to "two very certain tests" by which one could always distinguish man from a perfect and even living automaton. For one thing, though such an automaton could utter words and emit some responses to stimulation, "It never happens," warns Descartes, "that it arranges its speech in various ways in order to reply appropriately to everything that may be said in its presence, as even the lowest type of man can do." For another, it seemed to Descartes "morally impossible" that "there should be sufficient diversity in any machine to allow it to act in all the events of life in the same way as our reason causes us to act." But over and above these two arguments was the basic difference between the various material signs and motions producing sensory perceptions and the meaning connected with them. To see the *meaning* of signs and of events was in his view the privilege and exclusive ability of an immaterial, unextended mind. (ibid., 21-22)

McCorduck records that Descartes "argued that animals might lack the ability to think and the language to communicate to one another, but . . . they were not necessarily without consciousness, memory, emotion, or perception" (1979, 32), which proposition, together with his mind-body distinction, elicited objections from a group of theologians who "proposed that the beast-automaton notion can only lead its supporters to conclude that there is a continuity in intelligence between animals and human beings" (ibid., 32). Supporters lifted the notion of dualism above such objections.

None of Descartes' tentativeness was to be found in Thomas Hobbes, whom Jaki allows "fell captive to a flat theory of ratiocination in which all operations of the mind were traced back to addition and subtraction" (1969, 23-24); Hobbes, he observes, "greeted Pascal's arithmetic machine as the evidence of his views on human mind and thought. He described the brass and iron pieces of the machine as being invested with the functions of the brain and instructed to perform some of the most difficult operations of mind" (ibid., 24). Thus, by reducing human reason to a mechanical process and by ascribing to mechanical contrivances an artificial life, Hobbes sought to refute the prevailing notion of dualism.

Hobbes was "the first," Dreyfus contends, "to make explicit the syntactic conception of thought as calculation: 'When a man *reasons*, he does nothing else but conceive a sum total from addition of parcels,' he [Hobbes] wrote, 'for REASON . . . is nothing but reckoning.' It only remained," Dreyfus adds, "to work out the univocal parcels or 'bits' with which this purely syntactic calculator could operate" (1979, 69). This Gottfried Wilhelm Leibniz proposed to do.

McCorduck attributes Leibniz's fervent pursuit of an integral and differential calculus that would reduce "reasoning to an algebra of thought, a *calculus ratiocinator*," to his yearning "for a common language among scientists, so they could not only disseminate ideas, but also discuss them clearly and rationally" (1979, 33). Dreyfus suggests that Leibniz

> thought he had found a universal and exact system of notation, an algebra, a symbolic language, a "universal characteristic" by means of which "we can assign to every object its determined characteristic number." In this way all concepts could be analyzed into a small number of original and undefined ideas; all knowledge could be expressed and brought together in one deductive system. On the basis of these numbers and the rules for their combination all problems could be solved and all controversies ended. (1979, 69)

What Leibniz in fact found was the binary system of numbers; it remained for others to work out the algebra.

To the machinery of artificial intelligence Leibniz contributed a calculating machine, a refinement of Pascal's arithmetic machine, which could perform multiplication by rapidly repeating additions. Yet, though working from both directions that some present day researchers see converging in the realization of intelligent automata—devising a logical notation whereby human thought might be represented in a machine and creating a machine that might carry out logical operations of sufficient complexity to be classed as thought—Leibniz did not himself subscribe to such notions as were being put forth by Hobbes. "He took pains," Jaki records,

Formulation of an algebra of thought came out of the work of a number of early nineteenth century English mathematicians, among them Sir William Hamilton, Augustus De Morgan, and George Boole.

> to point out that a feeling or thinking being is not a "mechanical thing, like a clock or a mill." It was nonsensical to suppose, he [Leibniz] added, that "we might conceive sizes, figures and motions, the mechanical conjunction of which might produce something thinking and even feeling in a mass in which there was nothing of the sort, which thinking and feeling should cease also in the same way by the derangement of this mechanism." (1969, 25-26)

Consider, in light of Leibniz's remarks, this observation by a twentieth century researcher in artificial intelligence.

> Certainly you don't believe that neurons experience pain. Certainly you don't believe that the carbon, hydrogen, and oxygen that make them up experience pain. Yet science has told us the *incredible* and counterintuitive fact that if you take the ordinary molecules that are sitting around in this room and put them in the right configuration, they will feel pain.
>
> Now that's *crazy!* If anyone told you that, you'd have to laugh at them. But it's true. And there's a name for that in philosophy. It's the problem of emerging properties. How is it that something that you don't think of as having those properties can all of a sudden emerge with those properties? There's a mystery here, and it's a mystery that also applies to computers. (Rose 1984, 167)

Withal, Leibniz maintained the dichotomy between mind and body. He regarded these as "separate, but exactly matched, giving meaning to each other in a system of corresponding monads, clocks wound up to keep time together for eternity" (McCorduck 1979, 33).

Formulation of an algebra of thought was to come out of the work of a number of early nineteenth century English mathematicians, among them Sir William Hamilton, Augustus De Morgan, and George Boole. The latter, according to McCorduck, "laid the foundations of modern symbolic logic in *An Investigation of Thought on Which Are Founded the Mathematical Theories of Logic and Probabilities*. . . . published in 1854 . . . [and] commonly referred to as Boole's *Laws of Thought*" (ibid., 40).

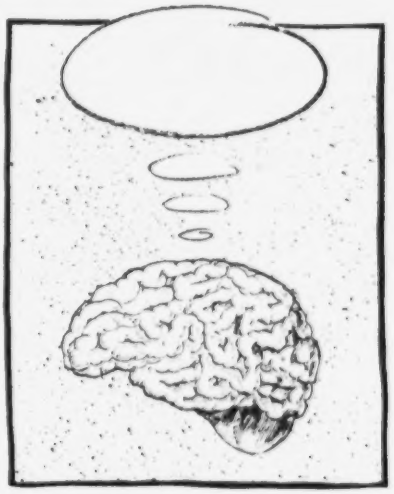
Though both are concerned with the general principles of reasoning, traditional logic employs words, whereas symbolic logic, to avoid the ambiguity that is inherent in natural language, uses ideographs. "Boole's system," McCorduck explains, "had 'elective symbols,' meaning arbitrary designations for classes of existing things, and 'laws of thought,' the rules of operation on these elective systems . . . which . . . significantly, would hold in an algebra of the numbers 0 and 1" (ibid.).

The earlier comparison of Oughtred's slide rule with Pascal's arithmetic machine identified the former as an analog, and the latter as a digital, device, the respective distinctions being the making of measurements of constantly varying quantities along a continuous scale versus the manipulation of discrete quantities. Because a logical algebra rooted in the binary system would, by definition, be implemented on the latter class of machine, an assessment of the progress being made by inventors of calculating machinery might be instructive.

Pascal's arithmetic machine, notes *The Annals of the Computation Laboratory of Harvard University, Volume I*, "was designed not to further scientific research but rather for use in his father's mercantile business. It was an accounting machine and as such was the forerunner of the modern accounting machine and cash register" (Harvard University 1946, 2).

A transitional phase ensued. "From the seventeenth century on," *The Annals* records,

> it was even more evident that precise and rapid methods of computation were required. The computation of tables of logarithms demanded by Napier's discovery of tables of sines and cosines, of tables of tides needed by faster and more extensive navigation and of the astronomical tables envisioned by Kepler, accentuated this need. Among many others, Gauss, Cayley, Tchebychev, Maxwell and Kelvin all attempted to devise or improve computational aids. . . . Despite this widespread interest . . . development of modern calculating machinery proceeded slowly until the growth of commercial enterprise and the increasing complexity of accounting made mechanical computation an economic necessity. Thus the ideas of the physicists and mathematicians, who foresaw the possibilities and gave the fundamentals, were turned to excellent purposes, but differing greatly from those for which they were originally intended. (ibid., 4)



The prototype of the modern digital computer, a machine conceived, but never built, by Charles Babbage in the early eighteen-hundreds, was successor to a somewhat, but *only* somewhat, given the relatively primitive state of machining at the time, less ambitious calculating machine Babbage called a "difference engine." According to *The Annals*, "this engine was to 'perform the whole operation-(the computation and printing of a table of functions)-without any mental attention when once the given numbers have been put into the machine" (ibid.). Although twenty years of labor would produce, beyond a first working model, only what Rose describes as "a pile of sprockets and cogwheels" (1984, 30), which are today preserved in the collections of the Science Museum in South Kensington, Babbage abandoned the project not out of frustration but rather in order to undertake the creation of a far more ambitious general-purpose calculating machine. Of Babbage's "analytical engine," *The Annals* observes that

> though the terms of the problem proposed were enough to stagger the contemporary imagination, he [Babbage] attempted to design a machine capable of carrying out not just a single arithmetical operation, but whole series of such operations without the intervention of an operator. The numbers in the first part of the machine, called the "store," were to be operated upon by the second part of the machine, called the "mill." A succession of selected operations were to be executed mechanically at the command of a "sequence mechanism". . . . For this latter, he intended

> to use a variation of the . . . cards. . . . used by the Jacquard weavers to control the looms to produce and reproduce the patterns designed by the artists. . . . Babbage required two decks; one of variable cards and one of operational cards. The first set was designed to select the . . . numbers to be operated upon from the store; the second . . . to select the operation to be performed by the mill. The . . . operation cards therefore represented the solution of a mathematical situation independent of the values of the parameters and variables involved. Thus the analytical engine was to have been completely general as regards algebraic operations. (1946, 5)

Interpretations of what Babbage thought of his designs vary. Jaki suggests that Babbage's appreciation for his computing device extended beyond its extraordinary capabilities to an anticipation that such devices would one day direct the course of science. But never, Jaki asserts, did Babbage credit his contrivances with the ability to think. He offers in support of this assertion his summary of a lecture Babbage gave in Turin in 1840 published in the *Bibliotheque universelle de Geneve*.

> In performing its calculation, Babbage remarked, the machine, unlike humans, cannot resort to any guesswork. It can necessarily admit only "the direct processes of calculation." Thus one of the earliest problems Babbage encountered in constructing his machine was to devise a method of division "without using the method of guessing indicated by the usual rules of arithmetic." This he had to do, convinced as he was that "the machine is not a thinking being, but simply an automaton which acts according to the laws imposed upon it." (1969, 47)

Indeed, Babbage, in *The Ninth Bridgewater Treatise*, gives an account of his machines as not only aids for calculation, but "a most impressive illustration of a train of thought indicating the existence of an infinite intellect, or Creator" (ibid., 43). Jaki suggests that Babbage viewed the Creator as an infinitely skilled programmer and considered that

> the immense complicatedness of the processes of nature. . . . was the result of the infinity of ways in which the material forces and entities in nature could interact with one another. . . . [which] implied in turn that a most extraordinary change might take place in the regular course of events at the most unexpected moments. To illustrate this, Babbage spoke . . . of the surprising turns that could be displayed by the calculating machine. . . . [which], constructed to follow a given law of calculation, might equally well be built in such a way as to begin to obey another law of calculation at a future time, known only to its constructor. The sequence of these two laws was evidently determined by another superior law, indicating the basic idea underlying the mode of operation of any advanced calculating machine. Thus the machine was in fact the embodiment of hierarchically ordered instructions or laws. As such, the calculating machine, so Babbage argued, was but a modest replica of nature. (ibid., 44-45)

Explications proffered by his friend and supporter, the Lady Ada, Countess of Lovelace and daughter of Lord Byron, are cited by Jaki and others as corroboration of Babbage's belief that his analytical engine was not a "thinking machine." The Countess, he explains, noted that "the Analytical Engine has no pretentions whatever to originate anything; it can do whatever we know how to order it to perform" (ibid., 48).

Acknowledging that "this statement has been adduced as evidence that machines cannot, in any way, be said to think," McCorduck contends that such an interpretation is misleading. "The countess," she points out, "prudently says that the actual existence of the machine and experience with its practical results would be the only way to answer questions about the machine's intelligence with any finality" (1979, 28).

It may be that the Countess' observations regarding Babbage's work are merely reflections of the latter's own perceptions, with little appreciable weight of their own; a recent review by Tracey Kidder (1985, 6) of a biography written by Dorothy Stein raises questions about the Countess' mathematical astuteness and suggests that she had considerable assistance from Babbage in the composition of a set of notes appended to a paper on the Analytical Engine by Luigi Menabrea which she translated from the French.

That the analytical engine suffered the same fate as the difference engine was due partly to the inability of machinists of Babbage's time to meet the required tolerances. Had it been realized, Rose projects, it would have comprised "a clanking assemblage of rods and gears" (1984, 30). Nevertheless, the unfulfilled design incorporated "the three essential elements of any modern computer: a 'store' (memory), a 'mill' (central processor), and a means of initiating various 'patterns of action' (programs")" (ibid., 30), and "the rods and gears of the feedback apparatus in Babbage's machine performed exactly the same function which is considered by several cyberneticians of our time to be the very essence of intelligent operation" (Jaki 1969, 46).

A number of less ambitious difference engines were built; *The Annals* records the successes of several Swedish and American inventors.

> In 1834, George Scheutz, a printer in Stockholm, built a . . . difference engine with the aid of a grant from the Swedish government. The machine was completed in 1853 and used for the computation and printing of tables of logarithms, sines and logarithms of sines. . . . From Sweden also came Wiberg's difference machine (1863) which was presented to the Academy of Sciences in Paris by the astronomer Delaunay.
>
> One of the first Americans to build a difference machine was G. B. Grant of Cambridge, Massachusetts, who needed a machine for his "computing for excavation and embankment." Encouraged in his design by Professor Wolcott Gibbs of the Harvard Mining School, Grant successfully built a small model in 1871 under Professor Benjamin Peirce, then superintendent of the Coast Survey. This machine was designed to contain the usual calculating and printing parts contained in Babbage's and

Julian Bigelow, Arturo Rosenbleuth, and Norbert Wiener created the science of cybernetics *out of their attempts to reconcile an apparent unity of problems in communication, control, and statistical mechanics with a lack of unity in research in these areas.*

> Scheutz' engines, but with considerable improvement in the printing mechanism. Grant's indebtedness for assistance in his study was expressed to John N. Batchelder of Cambridge and to Professors Eustis, Winlock, and Whitney of Harvard. (Harvard University 1946, 6-7)

The most profitable of the late nineteenth-century calculating machines was, according to Rose, "Herman Hollerith's automatic tabulating machine, an early data processor that was used to great advantage in the 1890 census. Like Babbage's unbuilt engine," Rose explains,

> the Hollerith tabulator used information supplied by punched cards; each card represented a person, and each punch-hole represented a piece of information about that person and allowed an electrical circuit to be completed that advanced a dial one notch. The success of this device enabled Hollerith to set up his Tabulating Machine Company. (1984, 31)

Finally, at the turn of the century, H. P. Babbage, with the advantage of electrical circuits and improved machine tools and materials of construction, completed a portion of his father's analytical engine. "A table of multiples of π which it computed to twenty-nine significant digits," *The Annals* records, "was published as a specimen of its work" (Harvard University 1946, 7).

In emphasizing the transfer of information rather than energy, cybernetics provided a new paradigm for understanding many different kinds of scientific phenomena.

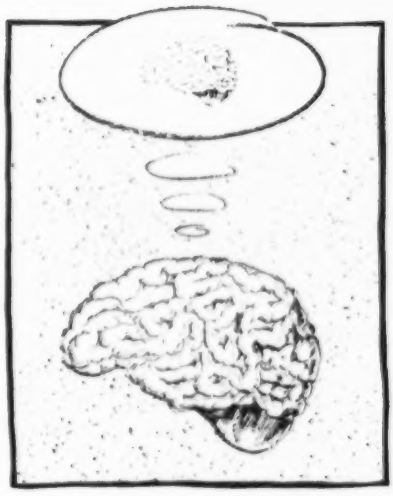
A contemporary of the younger Babbage, and himself an inventor of calculating machines and other automatic devices, Leonardo Torres y Quevedo built two electromechanical chess automata capable of playing the endgame of king and rook against king. "He declined," according to McCorduck, "to claim that his automata were actually thinking, but in 1915 he wrote, 'The inventor claims that the limits within which thought is really necessary need to be better defined, and that the automaton can do many things that are popularly classed with thought.' Torres," she continues,

> drew a distinction between the simpler sort of automaton, which has invariable mechanical relationships (the self-propelled torpedo was his example here; he had developed and demonstrated such a device) and the more complicated, interesting kind, whose relationships between operating parts alter "suddenly when necessary circumstances arise." Such an automaton must have sense organs, that is, "thermometers, magnetic compasses, dynamometers, manometers," and limbs, as Torres called them, mechanisms capable of executing the instructions that would come from the sense organs. (1979, 48-49)

McCorduck records that Torres' awareness of the general unreliability of electromechanical components and his lack of confidence in the ability of mechanical engineering to surmount the difficulties the elder Babbage had earlier encountered led him to view as dim the prospects for transforming his bolder designs into working machines. Nevertheless, she observes, "servomechanisms, which is what we would call Torres's machines, appeared in great numbers in the early twentieth century, and were the immediate inspiration for the field of cybernetics" (ibid., 49).

According to Jaki, it was "Vannevar Bush who rediscovered in the early twenties the principles embodied in Kelvin's differential analyzer and developed it into the first modern computer" (1969, 49). Dean of the school of engineering of the Massachusetts Institute of Technology (MIT) when he built the machine, Bush later assumed the presidency of the Carnegie Institution of Washington and, during World War II, served as director of the Office of Scientific Research and Development.

Jaki observes a growing susceptibility to the brain-computer analogy with the debut of the Bush Differential Analyzer; MIT's *Literary Digest* for 17 December 1927, he notes, variously heralded the device as a "man-made brain" and "an electrical machine which thinks for itself." Bush, rather less dramatically, described his construction as "an adding machine carried to an extreme in its design" (ibid., 49-50). Eight years later, in order to eliminate the need to rearrange the parts of the machine to solve different types of problems, Bush and his associates undertook the development of a computer with electrical connections.



In abandoning his difference engine in order to pursue construction of the more complex analytical engine, Babbage had made a conceptual leap from calculating device to general-purpose computer. But realization of his grander design wanted a higher technology; it was variations on the theme of his difference engine that populated the century following his abandonment of it. Then, in the early twentieth century, with technology on the ascendant, the concept of the general-purpose computer was resurrected in the work of Alan Turing, a mathematician graduated from Cambridge University, and Claude Shannon, an engineering student at the Massachusetts Institute of Technology.

Turing's contribution was an abstract universal computing machine which he introduced in a paper written to prove that certain classes of mathematical problems were not solvable by any fixed and definite process. Turing likely drew on his familiarity with Babbage's work in conceiving what has come to be known as the "Turing

(Continued on page 22)

If someone would doubt my results, I would say to him: "Let us calculate, Sir," and thus by taking pen and ink we should settle the question.

— Leibniz

Calculating and searching

Those who hold that a computer is nothing more than a fancy calculating machine have a point — on one level at least. Whether on another level the exceedingly rapid and extraordinarily complex calculation that takes place in a computer's bramble bush circuitry results in something akin to thought—conscious or otherwise—is a question that is entertained, with frequent reference to others who have entertained it, in the accompanying article.

The lead quotation from Leibniz reflects a millenia-old desire to reduce human thinking processes to a kind of precise calculation such as forms the basis of every modern computer's processing abilities. But while the computer has presented the long sought after structure, the means for transferring to it any but the most superficial forms of human thought has not attended it. Below a certain, relatively high, level we are not aware of our thought processes, whereas all we know of a computer's processing is confined to the very basic level of computational manipulation of electrical states. The problem this presents is summarized, rather succinctly, by one of the students whose work in artificial intelligence is recounted by Frank Rose in *Into the Heart of the Mind* (Harper & Row, New York, 1985): "When you educate a human, you don't have to figure out what they're doing inside their head. You just have to tell them things, and somehow they do all the remembering automatically. But here [in the computer] we have to figure out how things should be stored and how connections get made."

Trying to reconcile the way we think with the way our machines must, by their very nature, think has been perhaps the greatest problem facing those who would create a thinking machine in the likeness of a thinking human. The general tack for getting the two closer together has been to persist in the age-old pursuit of a consistent and precise method for representing human thought. The artificial intelligence community refers to the problem as that of knowledge representation.

In *Understanding Artificial Intelligence* (Texas Instruments, Dallas, 1985), Henry Mishkoff, in a chapter titled "Symbolic Processing," offers some insight into the nature of present day techniques for representing knowledge in computers. These schemes, he points out, owe much to the ancient Greeks who codified the system of human reasoning and deduction called logic.

Indeed, logic has found hearth and home in the computer, which, as we know it, is a much more logical system than any human who has ever lived. The principal problem with logic has been the intractability of much of human knowledge to representation in its rigid, formal structure.

What can be represented in logical notation are propositions, hence the system that, working from statements whose "truth values" (true or false) are known, can calculate the truth values of related statements is called *propositional calculus*. To illustrate his progressive definition of the logic employed in artificial intelligence, Mishkoff enlists the aid of the noted feline, Garfield. Propositional calculus, he explains, enables us, working from the following pair of statements whose truth value has been established as true:

- Garfield is a cat.
- If Garfield is a cat, then he is a mammal.

to infer for the following statement a truth value of true:

- Garfield is a mammal.

Predicate calculus introduces the ability to specify relationships and make generalizations. Again, Garfield demonstrates. Given known truth values of true for the following statements:

- Garfield is a cat.
- All cats are bigger than all mice.

we can, using predicate calculus, arrive at, and know to be true also, the conclusion that:

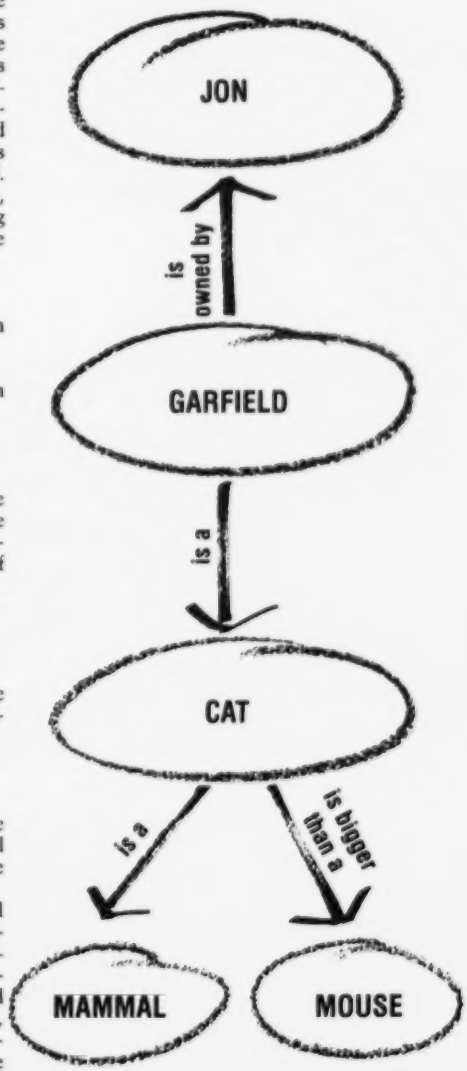
- Garfield is bigger than all mice.

The generalizations included in the foregoing statements are "all cats" and "all mice" and the relationship is "are bigger than."

Most often employed in artificial intelligence work is a variation of predicate calculus, called *first-order predicate calculus*, that extends the boundaries of logic to include functions and other analytical features. A *function*, Mishkoff explains, is a logical construction that returns a value. Thus we might have an "is-owned-by" function for Garfield that would enable an artificial intelligence program that incorporated it to respond to the question "Who owns Garfield?" with "Jon."

In order to further reduce the constraints on expression posed by logical notation, Lotfi Zadeh, a professor at the University of California at Berkeley, introduced the notion of *fuzzy logic*, a still less formal system that can accommodate such relative and approximate concepts as "tall," "expensive," and "normal."

Specifying highly complex interrelationships among knowledge elements yields a *semantic network*. A simple semantic network depicting the interrelationships of the facts we have regarding Garfield is shown in the accompanying illustration. The several



knowledge elements constitute *nodes*, whose *links* define their functional interrelationships. The characteristic that enables us to make deductions from this network, such as "A cat is a mammal and is bigger than a mouse; Garfield is cat; therefore, Garfield is a mammal and is bigger than a mouse," is called *property inheritance*. Within the context of the semantic network, Mishkoff explains, a cat has the properties of being a mammal and being bigger than a mouse, and since Garfield is a cat, he inherits those properties.

A form of logic was employed by what is generally considered to be the first artificial intelligence program, the Logic Theorist. But while expressing facts about, and relationships among, objects is amenable to logical representation, expressing knowledge about what to do with a collection of facts—the kind of working knowledge present-day expert systems must have in order to be able to recommend courses of action on the basis of analyses of the factual data available to them—is best accomplished by introducing a set of rules. These usually take the form of "if-then" expressions, in which the "if" (conditional) clause specifies a condition that must exist in order for the "then" (action) clause to be invoked. Such rules, (also known as *production rules*), are formal representations of the relevant heuristics, or "rules of thumb," by which we would ourselves evaluate and act upon the available facts. A personal heuristic of Jon's might be: *If* Garfield is rubbing against my leg and meowing plaintively; *then* I should take the necessary steps to feed him.

The structure of *rule-based* or *production* systems is advantageous for several reasons. Such systems can be modified relatively easily by incorporating new, or elmiminating existing, rules; their method of operation, which entails the successive "firing" of rules, makes building in accountability for their reasoning relatively straightforward; and, because knowledge in the form of such rules tends to be very readable, even a person who is unfamiliar with such systems can often use them.

Any artificial intelligence system that is required to reason its way to a conclusion must be able to search its store of accumulated knowledge rapidly and effectively. Little is known about how humans search their mental stores; much of the process seems to be informal and subsconcious. That humans know practically everything about how computer programs search their knowledge bases is attributable, quite unremarkably, to the fact that humans conceived the search strategies employed by computer programs.

Having conceded our possession of precious little knowledge about what goes on inside our heads, we must turn to what goes on outside of them for an analogy by which to explain some of the characteristics of searching as it is conducted by an artificial intelligence program.

Consider the prospect of travelling from Boston to Tokyo. To simplify the analogy, a westerly direction of travel is dictated. In the parlance of artificial intelligence, Boston is your *initial state* and Tokyo your *goal state*. All that lies between is your *search space*. An inverted tree structure, called a *search tree* is often used to depict a search space graphically.

Your search space could conceivably include every foot of highway and railroad track in this country and on the Japanese mainland, most of the vastness of the Pacific Ocean, and all of the airspace over that great body and two of the countries she separates. Considering the number of initial paths open to you, and the number of additional paths that open up at every fork in each of those paths, and at every fork in each of the paths that emanate from those forks, etc., will provide an understanding of what computer scientists are referring to when they discuss the problem of *combinatorial explosion*.

Your initial state, the goal state, and every fork in between, are *nodes*. If you were to undertake your journey by first travelling to each node that is connected directly to the node that is your initial state, then travelling to each node that is directly connected to each of those nodes, then travelling to each node that is connected to each of the nodes connected to those nodes, you would, again in the terminology of artificial intelligence, be conducting a *breadth-first* search. Such a search would not be practical for you, and it would not be practical for any program that is trying to reach a goal state that is separated from its initial state by many levels.

Alternatively, you could select one from the many paths directly connected to your goal state and, without looking back, select one of the available paths connected to each of the nodes that you encounter thereafter until you reach a *terminal node*, that is, a node with no further branches. The terminal node might be, but is not necessarily, the goal state and a trip to it in the manner just described is called a *depth-first* search. The disadvantage of a depth-first search becomes apparent when you consider that if the terminal node you reached was not the goal state, you have to start over from the beginning, and if the terminal node you reach at the end of that trip is not the goal state, you have to—, etc.

Working, as you have been, from your initial state to the goal state is termed *forward-chaining*. It is also possible to start from the goal state and work backward toward your initial state, which approach, it follows, is called *backward chaining*. You would very likely in your situation, and many artificial intelligence programs do, use a combination of forward and backward chaining to identify a route. This constitutes a *bidirectional search*. You would alternate from one to the other, but an artificial intelligence program can initiate simultaneous searches from both initial and goal states and terminate the process when the two seaches meet at a common node.

As performed, both of the searches described above—breadth-first and depth-first—were *blind* searches. Node-to-node progress was governed by nothing more than whether a given node corresponded to the goal state or not. This is termed the *generate-and-test* method. It would clearly be an impractical method for you and, when the search space is extremely vast, it is also an impractical method for an artificial intelligence program.

To the end of making searches more practical, ways of *pruning* a search tree have been devised. Just as you, in your Boston to Tokyo trek, would have recourse to maps and agents for air, bus, and rail lines, so an artificial intelligence program can be afforded various means to achieve a more direct transition from an initial state to a goal state. For example, the generate-and-test method can be enhanced through a technique called *hill-climbing*, whereby the program is provided with the ability to calculate the difference between the current node and the goal state and, by comparing computed differences at successive nodes, determine whether it is moving toward or away from the goal state. Having determined that it is moving away from the goal state, the program can look for a more optimal path with a minimum of backtracking.

Another method of search space reduction uses a bidirectional search strategy in conjunction with heuristics and subgoals. A program might, for example, have a heuristic that would lead it quickly to its goal state, but not be able to apply that heuristic from the current node. If, however, the heuristic could be applied from a nearby node, the program might set aside its ultimate goal in order to establish a subgoal of getting to the nearby node.

You might suspect that it will be a long time before an artificial intelligence program will be able to find the optimal route from Boston to Tokyo, but recently the author had an opportunity to talk with a program that, provided with his street address (initial state) and only a telephone number for his destination (goal state), was able to relate street-by-street directions from the former to the latter. *J.S.*

The field of information science *had its genesis in a 1943 paper by neurophysiologist Warren McCulloch and mathematician Walter Pitts that posited similarities between the operation of a network of neurons in a human brain and the operation of a computer.*

(Continued from page 19)
machine," but, though he did go so far as to describe its workings, he did not, as Babbage did, actually intend to build the machine. "Turing showed," Rose explains, "that such a device, operating according to explicit instructions expressed in binary code, would be a universal computer, capable of doing anything any other computer could do" (1984, 32), a concept, Rose avers, that was startling to mathematicians, implying, as it did, that "properly constructed and programmed, any one computer could in principle do the work of all computers. . . . [which], in turn suggested that a computer could be built that could do *anything* for which an algorithm—a precise and unambiguous set of instructions—could be written" (1984, 32). Known as Church's thesis (after Alonzo Church, a logician Turing worked with at Princeton), this latter proposition constitutes, Rose observes, "one of the cornerstones of computer science. . . . [and] the theoretical basis for artificial intelligence" (ibid., 32-33).

Contemporaneous with Turing's paper, Claude Shannon's masters thesis used Boolean algebra to describe the behavior of relay and switching circuits. "It supplied," according to Rose, "the link that connected binary math, symbolic logic, and the behavior of electronic circuits" (ibid., 32). Like Babbage's concept of a general-purpose computer, Boole's algebra lay dormant for a long time; some sixty years elapsed between his demonstration of the connection between mathematics and logic and the revival and perfection of the notion by Bertrand Russell and Alfred North Whitehead in their book *Principia Mathematica*. "Shannon's genius," Rose observes, "was to see that . . . the propositional calculus of symbolic logic could be used to describe the two-state, on-or-off behavior of an electromechanical relay switch. . . . and with this realization came the genesis of what has since come to be known as information theory—the idea that information, like energy or matter, is a quantifiable entity that can be manipulated at will" (ibid., 32).

Norbert Wiener, who took his Ph.D. from Harvard University at age eighteen, had, prior to the outbreak of World War II, shared with physiologist Arturo Rosenblueth a growing interest in analogies between electronic and biological devices. The interruption that was the War years found him paired with Julian Bigelow on a project to improve antiaircraft artillery.

Together, Bigelow, Rosenbleuth, and Wiener set about making a science out of the art of engineering design. They sought to reconcile with this new science the apparent unity of problems in communication, control, and statistical mechanics with a lack of unity in research in these areas. "They decided," McCorduck observes, "to call the entire field of communication and control theory, whether in the machine or the animal . . . *cybernetics*, from the Greek word for steersman" (1979, 45).

Supplementing their own knowledge about servomechanisms—mechanisms whose operation relies on some kind of feedback—with Rosenbleuth's familiarity with problems of excessive feedback in the human body, Weiner and Bigelow "devised a model of the central nervous system that explained some of its most characteristic activities as circular processes, emerging from the nervous system into the muscles, and reentering the nervous system through the sense organs" (ibid.). Their model, which emphasized the transfer of information rather than energy, provided a new paradigm for understanding many different kinds of scientific phenomena. Mishkoff explains that "by describing interrelated systems in terms of the exchange of information," cybernetics pointed out functional similarities between humans and machines which greatly contributed to our understanding of both (1985, 31).

Within a few years of the signal contributions of Turing and Shannon, Britain, Germany, and the United States hosted efforts to construct large computers. World War II lent impetus to some of these projects. Turing himself was associated with the British effort that yielded Colossus, an electromagnetic machine, completed in 1943, that provided a means to decode messages scrambled by a German automatic-encryption device called "Enigma."

By 1943, Konrad Zuse, a young German engineer apparently little heeded by his government, was, McCorduck relates, wondering of a calculating machine he had assembled in the parlor of his parents' Berlin apartment "whether it could play a master in chess." By 1945, she records, "he had developed a programming language . . . which, he felt certain, could be used for solving not only mathematical problems but also many other symbolic problems, such as chess moves" (1979, 50).

In the United States, 1943 saw completion, by Harvard University and International Business Machines Corporation (successor to Hollerith's Tabulating Machine Company), of the Automatic Sequence Controlled Calculator, or Mark I. This punched-tape driven calculating machine, which computed solutions to differential and integral equations using banks of telephone relays, was assembled under the direction of Harvard professor and Naval Reserve Commander Howard H. Aiken.

Commissioned by the War Department to compute trajectories for artillery shells, the Electronic Numerical Integrator and Calculator, or Eniac—a similar machine, except that it relied primarily on vacuum tubes rather than relay switches for representing binary operations—was constructed at the University of Pennsylvania's Moore

School of Electrical Engineering under the direction of John Mauchly and J. Presper Eckert.

That little quantitative data on the operation of the human brain existed in the mid-1940s was forthrightly acknowledged by a pair of the more thoughtful writers to suggest similarities between the brain and the computer. In 1943, Warren McCulloch, a neurophysiologist, with mathematician Walter Pitts, authored a paper that posited similarities between the operation of a network of neurons in a human brain and the operation of a computer. Henry Mishkoff notes that although the precise correspondences suggested by McCulloch were not borne out by subsequent enlightenment about the operation of the brain, "his persuasive descriptions of the similarity between minds and machines contributed a great deal to the development of a field called *information science* and eventually to artificial intelligence" (1985, 29).

In the absence of qualifying data from brain research, refinements to the computer proved highly suggestive to imaginative writers who were quick to generalize novel characteristics in computer operation to operations in the human brain. One such refinement was the *stored program* computer, a machine, Rose explains, that "carried its instructions inside instead of having to be laboriously reprogrammed for each different task" (1984, 34). First realized in the Edvac computer, and its British counterpart, Edsac, the stored program concept made it practical to use a computer in the preparation of its own programs and led to the development of compilers, operating systems, and other programming aids.

A resurgence of popular literature's indulgence in hyperbolic analogy accompanied the unveiling of these newer machines; their massive size prompted references to "giant brains," while the descriptions of other writers ran a gamut from allusions to antiquity, such as credited the machines with "oracular" powers, to more conservative, contemporary estimations that held them to be, for example, "educated machines."

John von Neumann was associated with the group that built the Eniac and, subsequently, the Edvac, and contributed significantly to, if he didn't wholly originate, the concept of the stored program. In 1945, in the specifications he was commissioned to write for a larger, faster machine for the University of Pennsylvania's Moore School, von Neumann made many explicit comparisons between the parts of the proposed computer and the human nervous system. But, like Leibniz's deep involvements with calculating machines and an algebra of thought, von Neumann's penchant for making analogies between the computer and the human nervous system did not translate into a conviction that they were in any way fundamentally similar entities. Jaki (1969, 79-80) cites several qualifications regarding brain-computer correspondences offered by von Neumann, among them:

- the number of actions that can be performed (he suspected that the human brain outstrips its artificial counterpart by at least four orders of magnitude);
- their respective components (whereas the brain utilizes a setup of numerous but slowly working components, a computer, he observed, employs fewer but faster units);
- the means of connecting their components (evidence suggested to him that the brain works mostly with parallel networks while computers are almost universally wired serially, which modes he regarded as not unrestrictedly interchangeable);
- their respective levels of precision (von Neumann noted that in the human nervous system a rather low level of precision, which he estimated not to exceed the level of two or three decimals, is associated with a very high level of reliability, which characteristic is not at all in evidence in computing machinery).

In his last public discussion on the subject, von Neumann suggested that "the nervous system appears to be using a radically different system of notation from the ones we are familiar with in ordinary arithmetic and mathematics. . . . Logic and mathematics in the central nervous system when viewed as languages, must structurally be essentially different from those languages to which our common experience refers" (ibid.). Of a detailed comparison, made by W. M. Elsasser, of the respective ways in which the two systems store information, Jaki notes

> his [Elsasser's] conclusions wholly corroborate von Neumann's views. . . . In particular, he noted in the brain the "amazing lack of evidence for the existence of devices that would store information mechanistically after the manner of the storage components of electronic computers." He also noted a . . . lack of evidence in the brain for associated devices on which the distribution and sorting of information might depend. (ibid., 80)

Such issues warranted wider attention and further discussion; to this end von Neumann, together with Aiken, who had just seen Harvard's Mark I to completion, Wiener, whose war work helped spawn the field of cybernetics, and Pitts and McCulloch, whose early paper positing similarities in the operation of the brain and the computer engendered the field of information science, formed, in December of 1944, the Teleological Society, its expressed purpose, to discuss "communication engineering, the engineering of control devices, the mathematics of time series in statistics, and the communication and control aspects of the nervous system" (McCorduck 1979, 66). Five years later, McCulloch addressed the first meeting of a similar organization, the Ratio Club, which had

The need for a forum for interdisciplinary discussion of issues associated with the emerging fields of cybernetics and information science led to the formation, in the United States, of the Teleological Society and, in Great Britain, of the Ratio Club.

formed in Great Britain. Among the founders of this organization were Donald MacKay, a physicist, and John Bates, a neurosurgeon. Alan Turing was a member.

Though they hadn't yet arrived at the term, the notion of artificial intelligence was being refined by members of these groups. McCorduck cites several relevant papers and creations by members of the Ratio Club. Grey Walter, she notes,

> the brain physiologist at whose laboratory the . . . Club was conceived. . . . made significant discoveries about the electrical activity of the brain, and . . . to study some of these further, he built the turtle, a dome-shaped electromechanical device that rolled its way around obstacles, and retired to its hutch when its batteries needed recharging. Walter intended it to show that complex behavior—what an observer might see as purpose, independence, and spontaneity—was in fact the result of rich interconnections of a relatively small number of original elements. (ibid., 81)

Another member of the Ratio Club, a psychiatrist named W. Ross Ashby, had long been certain that much human mental behavior could be accounted for mechanistically.

Ashby viewed organism and environment, taken together, as an absolute system whose parts interact with one another. Pondering the case for machines, he built one that McCorduck describes as "a self-organizing system, a system that responds to stimuli, changing its behavior, and in some sense its shape, in order to achieve stability" (ibid., 83). She describes this "homeostat" as a cluster of four units, each equipped with a governing device and capable of emitting a direct current output to the others and of receiving their output in turn. Relative to the settings of the governing devices, definite patterns of behavior would be exhibited by the homeostat as it sought to stabilize itself.

Ashby also contributed a pertinent observation to the evolving field of artificial intelligence: he pointed out that while both might be said to be designing a mechanical brain, the aim of one who wants his creation to perform a specific task, preferably better than a human can do it and not necessarily by methods humans might use, is substantively different from the aim of one who, like himself, is concerned with copying the living brain with such fidelity that the mechanical model would, in the same way that the human brain can, fail in certain characteristic ways. "Here," McCorduck notes, "Ashby articulated the distinction that would subsequently define two major branches of artificial intelligence: one aimed at producing intelligent behavior regardless of how it was accomplished, and the other one aimed at modeling intelligent processes found in nature, particularly human ones" (ibid.).

MacKay, during his years with the Ratio Club, "was wondering whether there was such a thing as the measurement of information, and in particular whether limits existed, in the ways that the Heisenberg Principle limits certain kinds of physical measurements" (ibid., 80).

McCorduck records that in 1949 MacKay privately circulated a document that discussed the possibility of combining analog and digital techniques in the design of an artefact that would be capable of responding to new information with adjustments in its behavior, that would, in effect, be capable of pursuing, independently of human intervention, a disciplined existence with at least an abstract sense of purpose and of developing purposes of its own that, while congruent with some overriding design principle, would not necessarily be predetermined by its designer.

In his own thinking about the notion of partial truth or probability, MacKay anticipated the present day concept of "fuzzy logic," an extension of more formal logic to accommodate terms of a relative nature, such as tall, brief, and nearby. "In a 1951 paper," McCorduck notes, "MacKay proposed a statistical mechanism that would display many of the attributes of human cognition, including prejudices, preferences, originality of a kind, and learning" (ibid., 81). Taking up, toward the end of that paper, the question of how mind arises from matter, MacKay concluded that it isn't the descriptions of mind or matter that are exclusive but the logical backgrounds in terms of which they have meaning, like the dual descriptions of light as waves and as particles, each of which is valid in its own context.

Meanwhile, Turing, in an article published in 1950, proposed what has come to be regarded as the classic test of machine intelligence. Although the test he suggested to provide the answer to his opening question, "Can machines think?" has come to be called the "Turing test," Turing originally proposed it as the "imitation game." The game was to involve three people, a man, a woman, and an interrogator. The interrogator was to remain in a room apart from the man and woman and, by putting questions to them, try to determine which was the man and which the woman. Turing labeled the three A, B, and C, C being the interrogator and A and B the man and woman. Whichever of the two is A is supposed to try to cause the interrogator to make the incorrect identification; whichever is B is supposed to help the interrogator. Turing suggested the use of teleprinters to handle communication between the rooms to eliminate tone of voice and handwriting as clues. The game becomes a test for intelligence in machinery, Turing proposes, when a machine is substituted for A.

In a paper describing his own work in artificial intelligence programming, conducted before it was even called by that name, Anthony Oettinger not only anticipated a difficulty of generalizing limited-domain programs that continues to be experienced to the present time, but also provided some insightful observations on using digital computers as models.

Of Turing's article, and particularly his test, Dreyfus remarks:

> The time was ripe to produce the appropriate symbolism and the detailed instructions by means of which the rules of reason could be incorporated in a computer program. Turing had grasped the possibility and provided the criterion for success, but his article ended with only the sketchiest suggestions about what to do next: "We may hope that machines will eventually compete with men in all purely intellectual fields. But which are the best ones to start with? . . . Many people think that a very abstract activity, like the playing of chess, would be best. It can also be maintained that it is best to provide the machine with . . . sense organs . . . and then teach it to understand and speak English. This process could follow the normal teaching of a child. Things would be pointed out and named, etc. Again I do not know what the right answer is, but I think both approaches should be tried." (1979, 73-74)

Both were. The same year that Turing's article appeared, an article was published by Shannon in which he discussed the options available to those who would program a computer to play chess. Shannon is credited, on the basis of this paper, with being the first to point out the impracticality of having a computer consider every possible combination of moves; he estimated that a computer that attempted to follow this approach, if it were capable of evaluating one million moves per second, would take 10^{95} years to make its first move.

Five years later, in a survey of the problems associated with writing chess-playing programs, Newell accompanied some suggestions for solving them with a proviso that the suggested mechanisms were so complicated as to make it impossible to predict whether they would work. Notwithstanding his cautionary hedge, a group at Los Alamos succeeded in programming a computer to play a poor but legal game on a reduced board the following year.

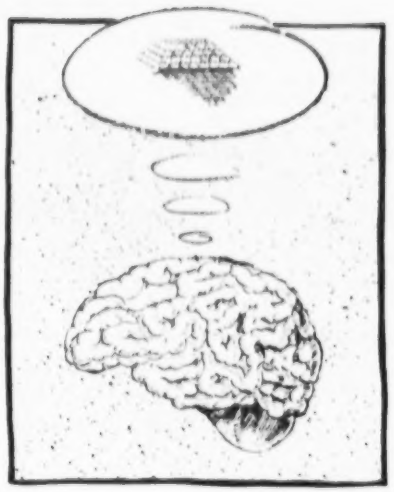
Although the Los Alamos group's program hadn't been written yet, chess-playing programs had been proposed when Arthur Samuel, in 1947, decided to build a checkers-playing machine. Samuel chose checkers specifically because it seemed a more tractable pursuit than chess, and the project was, for him, peripheral to his principal aim; the checkers-playing machine was intended to play in, and win, a checkers tournament gaining thereby notoriety and, hopefully, raising money for the construction of a computer on the scale of the Mark I or Eniac for the University of Illinois. Not only was the program, (not to mention the very small computer it was to run on), not completed in time for the tournament he had hoped it would play in, but Samuel was to work on the program, off and on, for some twenty years more, part of which time he would be employed by International Business Machines Corporation (IBM), the company that had earlier cooperated in the creation of Harvard's Mark I.

Many of IBM's early data processing computers cut their teeth on Samuel's checkers program, which, because it was complicated and long-running, proved an excellent test vehicle.

Though too late for his Illinois venture, notoriety did begin to accrue to Samuel's program, somewhat to the chagrin of conservative IBM, which didn't mind using it to test computers, but worried lest it nurture public fears about thinking machines and computers taking over the world. Samuel's program was becoming increasingly sophisticated; ultimately, it could, to some extent, learn, supplementing the look-ahead approach to making moves with recollection of consequences of moves made in identical situations in previous games.

Samuel's approach to imparting learning ability to his program could very well describe the approach employed today by those working in the branch of artificial intelligence concerned with the development of expert systems.

> He did not believe that human learning processes ought necessarily to be imitated, because he believed that the differences between the human brain and the computer were simply too great. "I think you study the way people solve problems to get an insight into what the real problem is, not to get an insight into the method the brain uses to solve the problem. And then you sit down and you say, 'Okay, given the technology available, the speeds, and what the computer will do, how best can we solve the problem?'" (McCorduck 1979, 152)



Anthony Oettinger, then at the University Mathematical Laboratory, Cambridge, Great Britain, tried a variation on the latter approach suggested by Turing, that is, teaching a computer as one might a child. One of his programs, which he dubbed his "shopping programme," divided the University's Edsac computer into two parts; one part of the machine became the shopping environment, the other part the shopper. Oettinger summarized the program thus:

> The shopping programme may be imagined to define a small child sent on a shopping tour. If this child were asked by its mother to buy different articles in various shops, it would not know at first where to find these articles, and would hunt for them by going from shop to shop in a random fashion until it came to the desired one. Having found an article once, it would remember in what shop, and would go directly to this shop the next time the same article were requested. In addition, as its curiosity would prompt it to note the whereabouts of articles for which it had not had a specific request, it would often be able to go directly to the right shop. (Oettinger 1952, 1247)

Oettinger also developed a response-learning program that was capable of adjusting its responses, on the basis of feedback provided by an experimenter, to stimuli of varying intensity detected by a sensory device. The experimenter could, by rewarding, punishing, or simply ignoring particular responses made

by the machine, cause it to tend toward certain responses over others. In contrasting the two programs, Oettinger suggests that the shopping program is "capable of performing functions which, in living organisms, are considered to be the result of intelligent behavior," whereas the response-learning program "is operating at a level roughly corresponding to that of conditioned reflexes" (ibid., 1251).

Oettinger admitted some severe limitations to the shopping program. These included limitations in the size of the matrix by which he defined the shops and their stocks, and the explicitness of the method by which the matrix and the memory were scanned. That Oettinger, in his shopping program, was thinking of emulating human thought processes rather than realizing intelligent operation by any means is clear when he suggests, by way of improving his program, "the introduction of memory decay or forgetfulness, [which] would give the learning process a greater degree of verisimilitude, that is a greater resemblance to the mental operations to which we are accustomed" (ibid., 1250).

Oettinger also recognized early, in his own programs, a characteristic that was to become endemic to artificial intelligence programming—that of limited domains. Commenting on the desirability of generalizing his shopping program, Oettinger observes that "while the shopping machine appears to learn, it certainly cannot be taught any subject the experimenter chooses" (ibid.). Describing a scanning method that would generalize the shopping program, Oettinger remarks that it would yield a machine that "is more powerful than the simpler machine only in the sense that its limitations are shifted to a higher level of abstraction." He goes on to observe that

> making it possible to teach the shopping machine the scanning method which it must use in turn to learn the contents of the shops is only one of many imaginable generalizations. . . . Additional generalizations, and generalizations of these generalizations, could broaden the range of subjects which can be taught to the . . . machine, but in every case the method of operation . . . at the highest level of generalization would have to be prescribed, and the range of operation consequently limited. (ibid., 1251)

Alan Newell, Herbert Simon, and J. C. Shaw created what some consider to be the first working artificial intelligence program in the Logic Theorist, whose solutions to mathematical problems and puzzles seemed intelligent and creative, and whose implementation techniques made significant contributions to the art of programming.

Oettinger's paper, written before artificial intelligence programming was called by that name, not only anticipated a difficulty of generalizing limited-domain programs that continues to be experienced to the present time, but also provides some insightful observations on using digital computers as models.

> It is important to distinguish carefully between machines intended to be models of the physical structure of animal nervous systems and machines designed to perform specific functions of animal nervous systems. The fact that . . . machines . . . can play restricted versions of the imitation game implies a correspondence between their functions and some functions of the human nervous system. But as there are usually many means of synthesizing a given pattern of functions it is dangerous to reason from functional to structural correspondence. A special machine could be built, for example, to do only the job of the shopping . . . machine. The physical structure and the internal mode of operation of this machine would need to bear no relation to the Edsac and, in turn, to the human brain, other than that of superficial correspondence of certain functions of input and output. (ibid., 1261)

Oettinger subsequentiy, Dreyfus notes, "turned to the problem of language translation and programmed a Russian-English mechanical dictionary. Further research in these directions, it seemed, might lead to a computer which could be taught to associate words and objects" (1979, 75).

But neither of the approaches outlined by Turing—abstract activities like game-playing or the teaching-learning paradigm which regarded the computer as an infant to be acquainted with the world—offered anything like a general theory of intelligent behavior. "What was needed," Dreyfus remarks,

> were rules for converting any sort of intelligent activity into a set of instructions. At this point, Herbert Simon and Allen Newell, analyzing the way a student proceeded to solve logic problems, noted that their subjects tended to use rules or shortcuts which were not universally correct, but which often helped. . . . [and] decided to try to simulate this practical intelligence. The term "heuristic program" was used to distinguish the resulting programs from programs which are guaranteed to work, so-called algorithmic programs which follow an exhaustive method to arrive at a solution, but which rapidly become unwieldy when dealing with practical problems. (ibid.)

The term "heuristic" the two borrowed from George Polya, a mathematician whom Newell had studied under at Stanford University, and whose attempts to demystify problem solving techniques used by mathematicians had earlier been gathered together in a book called *How to Solve It*.

While at Rand Corporation in the early 1950s, Newell had applied card-programmed calculators to the simulation of radar screens, an activity, when he became acquainted with it, that "fascinated Simon, and opened his eyes to the possibilities of a computer as an information processor" (McCorduck 1979, 129).

When Newell and Simon came together again two years later, in 1954, they were in turn influenced by a third individual, Oliver Selfridge, who had come to Rand to describe his work with G.P. Dinneen on pattern recognition. Selfridge and Dinneen had developed a program that attempted to recognize visual patterns—a few alphabet letters and some simple geometric figures—by computing characteristics of the figure and comparing the returned values with norms. But more significantly, the program pos-

sessed the ability to generate new characteristics and to eliminate characteristics that it tried unsuccessfully, in effect, to learn from its experience. "It didn't learn very well," McCorduck notes,

> indeed, it tended to fix itself on one characteristic and improve that one to the exclusion of others. But that wasn't the point. . . . the thing that had so taken Newell . . . was that a complex process was underway that was the result of many simpler subprocesses. . . . [which] had been organized in a highly conditional and interactive way, and the system showed that, working in concert, a set of simple subprocesses that were easy to understand could lead to genuinely intelligent behavior. This conclusion seems altogether obvious and common sensical in retrospect, but at the time it wasn't at all obvious and it was counter to common-sense notions of intelligent behavior. This assumption—that sets of simple subprocesses could produce a system which behaved in a complex way—was to inform research in artificial intelligence through many, many more tasks of greater and greater difficulty. (ibid., 134)

The effect of this on Newell, McCorduck explains, was to turn him from theorizing about how human organizations accomplish tasks to trying to build systems that exhibit intelligent behavior. His first step in this direction, she reports, was to write a paper "using chess as a vehicle to understand what Selfridge and Dinneen were doing" (ibid., 135). To carry his intention from paper to program, Newell left Rand for the Carnegie Institute of Technology (later Carnegie-Mellon University) to seek Simon's collaboration. The work Newell purposed to do

> was altogether congruent with Simon's growing convictions that the computer could be made to simulate human thought processes in ways that would yield insights which previous models—mathematical, statistical, behavioral—had not. The result would be the information-processing model, child of the new scientific paradigm of information theory. (ibid.)

The two subsequently determined to create not only a chess machine, but also a program to prove mathematical theorems from *Principia Mathematica*. Work on the latter program, which Newell and Simon would call the Logic Theorist, eventually preempted work on the chess machine, and also came to include J. C. Shaw, a senior programmer at Rand, whose expertise was tapped remotely via long-distance telephone. In the closing days of 1955, these three were able to lay claim to a working Logic Theorist whose solutions to mathematical problems and puzzles seemed intelligent and creative, and to some significant contributions to the art of programming.

Memory, which is still a precious commodity in computers, was no less so in those early days, and computers then, as now, typically allocated it at the beginning of a program's run, irrevocably dedicating segments of storage to specific functions. So great was their Logic Theorist's appetite for memory that the Newell, Shaw, and Simon team was forced to come up with an alternate approach, which consisted in labelling each segment of memory and providing the computer with the means to keep track of the usable space available at any given time. It was from this "vacancy list" that their *list processing* technique took its name.

List processing also introduced to programming the notion of *recursion*, which describes the ability of a program to handle instructions whose execution is contingent on the execution of one or more instructions earlier in a sequence. The notion of list processing carried into the organization of data, as well. "It happens," McCorduck explains, "that data arranged in lists, and lists of lists, lend themselves most readily to the simulation of human thinking processes. . . . [which] fact is unsurprising, for . . . one important source of the ideas for list-processing languages was what psychologists knew about associative memory in human beings" (ibid., 143).

A modest gathering in the mid-1950s, commonly referred to as the Dartmouth Conference, served to collect briefly, in one place, the handful of individuals who would for several decades lead research in a new discipline, and gave that discipline the name of artificial intelligence.

Approximately twenty-five hundred years interceded between Socrates vision of reasoning reduced to a form of calculation and the vision of a group of individuals who sought to focus through the medium of the digital computer what had amounted to little more than a mirage for Socrates. This later vision took the form of the Dartmouth Summer Research Project on Artificial Intelligence (often referred to as simply "The Dartmouth Conference"), a two-month event made possible in part by a grant from the Rockefeller Foundation and attended by a core of some ten mathematicians, psychologists, neurologists, and electrical engineers from industry and academia. Four of the ten collaborated on the grant proposal which called for

> a . . . study of artificial intelligence [to] be carried out during the summer of 1956 at Dartmouth College in Hanover, New Hampshire. The study is to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it. (ibid., 93)

John McCarthy, an assistant professor of mathematics at Dartmouth and the conference's prime mover, drew the other three collaborators from associations past and present. Marvin Minsky, a junior fellow in mathematics and neurology at Harvard University, had been a graduate student with McCarthy at Princeton University and the two had subsequently worked at the Bell Telephone Laboratories with Claude Shannon. Nathaniel Rochester, Manager of Information Research for IBM, was known to McCarthy in connection with the gift of a computer that his company was making to the Massachusetts Institute of Technology; these two discovered a mutual interest in intelligent machines, particularly in such machines as might be made to exhibit original behavior in problem solving.

Of the other conference participants, some have already been associated with ideas or lines of work—Samuel, Selfridge, Simon, and Newell. Also present were Alex Bernstein, from IBM, who, in developing a chess-playing program that employed sophisticated heuristic techniques for identifying best moves, was exploring various methods of eliminating possibilities, a concept that has become central to artificial intelligence research, and Ray Solomonoff, who was working on models of inductive inference and arguing the case for presenting simple problems to computers in order that the method by which the machines arrived at their solutions, and not the solutions themselves, might be more carefully scrutinized.

Reviewers seem to agree that the Dartmouth Conference is more noteworthy for the individuals it brought together than for anything that transpired in the course of it. McCorduck remarks that it represented a confluence of several different intellectual streams of the twentieth century which had, themselves, flowed from other streams (ibid., 95), and Mishkoff adds that in this confluence a new branch of computer science arose, the term *artificial intelligence* becoming firmly and irrevocably associated with it (1985, 32). "There were no spectacular scientific developments at the conference," he observes, "no startling insights. . . . but the new field did come away with a name and with an "elite". . . . [and] even today, the leadership of the American artificial intelligence community is composed largely of the conference participants, their students, and their students' students" (ibid.).

So it was that, though some of them used it not, they had a word and, going forth, these few drew to them others and so the word was spread. Rose names as the early temples of artificial intelligence Carnegie-Mellon University, the Massachusetts Institute of Technology, and Stanford University, the schools at which such conference luminaries as McCarthy, Minsky, Newell, and Simon settled.

At Carnegie-Mellon, where they had developed the Logic Theorist, Newell and Simon pressed on with the development of a successor program called the General Problem Solver, or GPS. In his tutorial work, *Artificial Intelligence*, Patrick H. Winston summarizes GPS thus:

> For some problems, it is natural to think in terms of current situation and desired situation. We speak of the collection of facts that specify the problem and where we are on it as the *current state*. Similarly, where we want to be is the *goal state*. Here are some examples:
>
> - In a travel problem, the current state and the goal state are defined by physical locations.
> - In a robot-assembly problem, the current state and the goal state are defined by the raw materials and the thing to be assembled.
> - In a geometry problem the current state is all that is known, both general and specific. The goal state is all that is know, as before, but also including the fact to be shown.
>
> Consider an approach to these problems in which *procedures* are selected according to their ability to reduce the observed *difference* between the *current state* and the *goal state*. This approach is known as *means-ends analysis*. GPS is a metaphor denoting a particular control strategy built on top of the means-ends analysis idea. (1984, 146-147)

In GPS, Newell and Simon did not, McCorduck avers, invent a new universe, but rather conceived a means for understanding some major aspects of an existing universe, that of human symbolic behavior. They associated their program with human problem solving, which they identified as "a subspecies of thinking, concerned explicitly with the performance of tasks" (1979, 213). McCorduck notes successful trials of GPS with logic problems and puzzle solutions, and incorporation of its principle techniques in a host of successor programs.

GPS was not alone in providing insights. At other institutions, other targets were being hit with varying degrees of success. John McCarthy, who had left Dartmouth to help establish an artificial intelligence laboratory at MIT, was aiming at generalized problem solving with a twist; he proposed a program that would be able not only to solve a variety of problems, but also to take advice in order to improve its performance. McCorduck records that this Advice Taker program never became anything more than a proposal "because," according to McCarthy, "in order to do it, you have to be able to express formally that information that is normally expressed in ordinary language" (ibid., 218). McCarthy considered this to be the key unsolved problem in artificial intelligence and made several unsuccessful attempts to invent a formal language that would be able to express the events of everyday life. Lack of success in this endeavour—his or otherwise—McCarthy attributed to too few competent people having devoted themselves to it.

In addition to aligning another target for artificial intelligence researchers, McCarthy's specifications for the Advice Taker engendered the development of two tools—the LISP programming language and the concept of timesharing—that have become fundamental to work in the field, as well as, to some extent, to computing in general.

The language and subsequent versions thereof that derived from the list processing technique developed by Newell and Simon in the creation of the Logic Theorist were called Information Processing Languages, or IPLs. The programming language McCarthy developed to accommo-

Although it was never realized, a program proposed by John McCarthy that would be able not only to solve a variety of problems, but also to take advice in order to improve it performance, yielded two by-products that have had great significance for artificial intelligence and computing in general—timesharing and the LISP programming language.

date the specifications of his Advice Taker program he called LISP. Minsky, in *The AI Business*, suggests that, at twenty-five years of age, LISP is still in its prime. "One of the things that we want in intelligent systems," he observes,

> is the ability to solve systems of complicated constraints automatically, but I am not sure that is the kind of thing you want to put into a language. The reason why LISP has retained its popularity in Artificial Intelligence is that it is not a language so much as it is a language that you write your own language in. When an artificial-intelligence person wants to make a program to reason by analogy, he starts with LISP because in a week or two he can write in LISP the elements of another language that he really wants to use. In PASCAL you just do not go around writing another language that you would rather have than PASCAL because PASCAL's syntax is too rigid and the way it allocates memory is too inflexible. LISP is really the machine language of high-level languages. (Winston and Prendergast 1984, 253)

Roger Schank, who approached the problem of knowledge representation on a computational-linguistic bearing, developed a series of programs that were able to digest and then answer questions about stories inferring information not explicitly contained in them.

Mishkoff describes some of the characteristics and functional attributes of LISP. One of the former is the list-orientation of the earlier IPLs. LISP lists comprise sequences of *elements*, each of which can be either an *atom* (a single object) or another list. A list can also be a sequence of lists. A term that is part of its own definition, as *list* is, is said to be *recursive*. What recursion allows in the present context is a program to arrive at a solution to a problem by working back through a series of antecedent problems, each of which depends for its solution on the solution of the preceding one. This technique, which is not supported by many programming languages, is regarded as an important component of intelligent problem-solving.

Data, which Mishkoff describes as *declarative* knowledge, that is, knowledge about the properties of an object, and instructions, which he defines as *procedural* knowledge, or knowledge about what actions to perform, take the same form in LISP (1985, 156). This consistency makes integration easy and enables LISP programs, which are themselves essentially lists, to be treated as data by other LISP programs, or even by themselves. Derivatives of this capability include LISP programs that can modify and add to their own instructions, and even write complete LISP programs, all of which is of great utility to a program that is attempting to learn a new task. LISP programs can also easily keep track of the order and frequency of the instructions they execute, an attribute that contemporary "expert systems" use to advantage in accounting for their actions to their programmers or users.

Although it is today the single most widely-used language for developing artificial intelligence programs, "it happens," McCorduck explains, "that LISP didn't catch on for some time. McCarthy," she says,

> attributes its late blooming to the fact that it could do things powerfully ... but at the time of its invention, nobody really wanted to do them. The simple programs most people were aspiring to were actually easier to program in machine code, and not until aspirations rose did people realize that LISP existed and would provide a representation by which they could accomplish more complicated tasks. (1979, 216)

The extensive use it received when once it did catch on led to the creation of total programming *environments* around LISP. These environments comprise sets of programs—including editors (for writing and modifying programs), debuggers (for finding and correcting errors in programs), and windowing systems (for displaying multiple sections of programs simultaneously)—that work fluidly with particular versions of LISP to expedite program development.

McCarthy's design for LISP included the notion of "working with it interactively—giving it a command, then seeing what happened, then giving it another command" (ibid., 217). To accommodate this approach, he was compelled to invent a complementary kind of computing, earlier pondered but never implemented by Shaw, which McCarthy was to call "time-sharing," after a term used in communications to describe a technique for sending several signals over the same line.

McCarthy's implementation of timesharing, which he carried out at MIT with the aid of a grant from the National Science Foundation, enabled several programmers to execute their programs interactively on a single computer. As with transmission systems, processing speeds of computers are so fast that, although the machines cycle serially among the programmers who are using them, the effect is of simultaneous processing. This was in sharp contrast to the then-prevailing mode of computing, termed *batch processing*, which called for programmers to read into a computer stacks of cards into which their program instructions had been coded in the form of punched holes, and then wait for the instructions to be processed and the results of the processing printed before making changes or additions and repeating the entire process. "It was," McCarthy remarked,

> one of those ideas that seemed inevitable in the sense that when I was first learning about computers, I was a little surprised that even if that wasn't the way it was already done, it surely must be what everybody had in mind to do eventually. It turned out it wasn't, and I promoted it as something for artificial intelligence." (ibid.)

In the mid-1960s, McCarthy left MIT for Stanford University, where he initiated a project in robotics research and today continues his work on the formalization of common sense knowledge and reasoning.

Minsky was McCarthy's collaborator in the creation of the MIT artificial intelligence laboratory and served for several years as its direc-

In the early 1970s at the Stanford Research Institute, a working model of a robot named Shakey that "understood" a highly constrained environment sufficiently well to formulate and carry out simple plans within it was built.

tor. His early work in artificial intelligence, influenced by the brain-computer comparisons of McCulloch and Pitts, was concerned with trying to understand the operation of the brain in terms of neurons using the digital characteristics of the computer. "Prodded," according to McCorduck, "by his colleague, Ray Solomonoff, who complained that there must be a more direct way of working on ideas of intelligence, such as trying to define what the behavior was instead of the parts it comprised, Minsky moved away from the neural-net idea" (ibid, 84), but not before contributing something of a last word on one of the major efforts to utilize this approach—the Perceptron.

Central to the notion of a neural-net is the assumption that organic brain cells are largely undifferentiated and organize themselves into purposeful behavior as a consequence of experience and perception. The built-in redundancy in the brain—by which responsibility for functions carried out by a part that becomes damaged can sometimes be assumed by another part—lends support to this notion, as does the seemingly random way in which connections are made between neurons, enabling impulses to detour when a route is damaged or otherwise blocked.

The Perceptron was an attempt to create an automaton that would be able to demonstrate equivalent capabilities, that is, that would be able to organize itself, to do so in a random manner, and to learn and adapt in the process. Work on the Perceptron was led by Frank Rosenblatt at Cornell University. As originally conceived, the system comprised three levels: corresponding to the retina of the eye was a grid of photocells that reacted to light stimulus; below that, a group of associator units collected impulses transmitted from the photocells to which they were randomly wired; and at base, a set of response units acted upon the signals received from the associator units. Subsequently, to improve its performance, some of the randomness of the Perceptron's wiring was eliminated on the assumption that animals and humans enter the world with at least some neural connections already established.

Minsky's final word on Perceptrons took the form of a book on the subject which he coauthored with Seymour Papert, whom he met in 1961 at a conference in England while presenting some work on a Perceptron-like machine.

"One problem with a Perceptron," McCorduck notes,

> was that it could classify stimuli it received but it lacked an internal representation of that act. Therefore it couldn't refer in some symbolic way to the act of perception, but had to recapitulate the act itself exactly, which put it in the position of being no better off when repeating the act again. Symbolic representation, by which machines (and humans) deal with a host of phenomena without having to reiterate them in their totality, is central to intelligent behavior, to memory, and to consciousness. (ibid., 89)

In turning his attention from brain structure to brain process, Minsky was drawing a bead on what has perhaps been the most persistent and intractable problem in artificial intelligence—knowledge representation. "We don't have to think about how to store what we know about love, say, or the behavior that's expected of us in restaurants," Rose observes, "we either store it or we don't." But for computers, he explains, "it's different: every fact . . . must be written out in code, and it must be organized in such a way that the computer can integrate it with every other fact it has" (1984, 55). It was to this end that Minsky introduced the notion of storing knowledge in *frames*. Within a given frame, a host of details is associated with a particular object or concept. A frame for a rose, for example, might contain details about color, number, fragrance, etc. When we think "rose" we typically assume default values for many of these details unless our immediate experience leads us to substitute different ones. For example, we might, at the mention of the word, call up the color red, only to change this image when the speaker presents us with a yellow rose. Details can be added to or eliminated from frames at will.

Roger Schank came to the problem of knowledge representation by way of language understanding. Donald Michie, who had earlier worked at Bletchley Park where the project that yielded Colussus was carried out and then moved to Edinburgh University where he became involved in artificial intelligence research, notes that

> in the 1950s and 1960s, millions of dollars were spent in the United States on research-and-development projects aimed at . . . ["machine translation"]. The techniques of machines breaking up texts grammatically and looking up meanings in a computer dictionary proved too shallow to crack the machine translation problem unaided. Fundamental progress had to wait for the development of an adequate theory of what is involved in "understanding" a passage of English-language text. (1974, 103)

Schank was one who attempted to construct such a theory, applying to the problem a computational-linguistic approach that assumed that ambiguity could be minimized if language could be represented at a sufficiently deep level. At such a deep level, he reasoned, different sentences with the same meaning would take the same form, while similar, or even identical, sentences with different meanings would register those differences. Because Schank assumed that primitive units of meaning could be identified and were related to one another by dependencies, his view, which was similar to that of noted linguist Noam Chomsky, came to be described as the notion of "conceptual dependencies."

Later, with Robert Abelson at Yale University, Schank attempted to integrate these concepts into larg-

er contexts. Out of this work came the notion of *scripts*. Like a frame, a script is a sort of atomic unit. It provides a set of default behaviors related to common situations. The script idea suggests that for almost any situation that we find ourselves in we have stored a pattern of behavior that is appropriate to it and that can be readily called up. As with frames, the elemental details of scripts can be augmented or reduced at will and possess default values that can be altered on the basis of present experience.

Terry Winograd, a doctoral candidate at MIT in the early 1970s, created a "bodiless" robot in software, called SHRDLU, that worked in an imaginary world populated by blocks of various shapes and sizes.

More recently, Schank has been trying to effect in programs the ability to make inferences. His resulting story-understander programs have demonstrated the ability to digest and then answer questions about stories inferring information that isn't explicitly contained in them. Linking a number of these programs together, Schank created BORIS (for Better Organized Reasoning and Inference System), a composite entity with a limited understanding of, among other things, what motivates people.

News stories are routinely digested by story-understander programs developed by Yale students. One such program, called IPP, has demonstrated an ability to learn and make generalizations from news stories about terrorism, such as that Italian terrorists tend to kidnap businessmen and IRA terrorists tend to send letter-bombs.

Schank's naming conventions—as evidenced in the terms that comprise BORIS, and the phrase Script-Applier Mechanism, from which he derived the acronym for one of his earlier story-understanding programs, SAM—are conservative, and perhaps appropriately so when compared to the terms concatenated by earlier researchers, such as Newell and Simon, whom Dreyfus takes to task for the audacity of their claim to having realized a General Problem Solver in GPS.

Randall Davis, in *The AI Business*, summarizes the early apirations, and subsequent realizations, of those working in the artificial intelligence community.

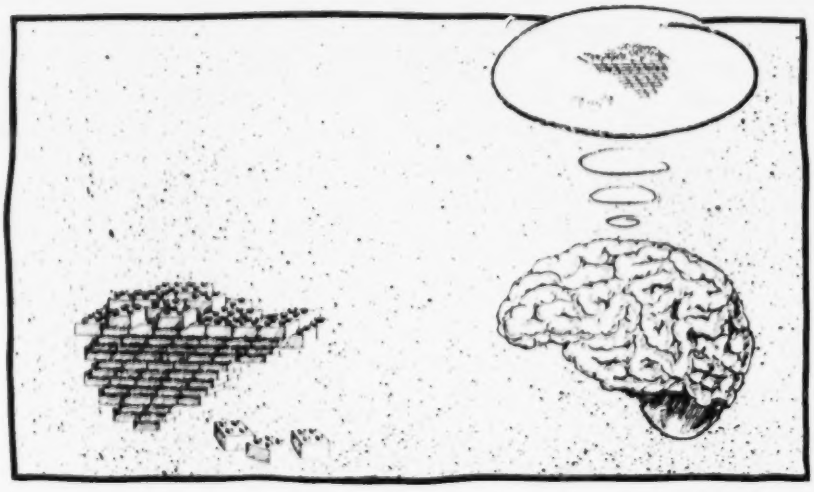
> In the early 1960s workers in the field thought if they could find one or two powerful ideas, they would be able to solve the problem of Artificial Intelligence. Since generality seemed to be a key aspect of human intelligence, research on generality followed. . . . The approach . . . did not produce the progress we expected. . . . A person can do something with almost every problem . . . [but] will do terribly on almost every problem, except the ones he . . . understands. . . . People solve problems well only when they know a great deal about the problem domain. . . . Our programs will have to know what they are doing. (This sounds obvious today, but it was an insight that took a substantial effort to establish firmly.) Expertise includes a constellation of behaviors. . . . Problem solving is . . . obvious and . . . necessary . . . [but] insufficient. Would we be willing to call someone an expert if he could solve a problem but was unable to explain the result, unable to learn anything new about the domain, and unable to determine if his expertise was relevant? (Winston and Prendergast, 1984, 19)

One of Schank's students at Yale, Robert Wilensky, earned his doctorate trying to expand the skills-repertoire of story-understanding programs. "While most of his [Schank's] students," Rose explains,

> were working on his main story-understander program, SAM, Wilensky was put to work on a more speculative cousin known as PAM, for "Plan-Applier Mechanism." Where SAM was supposed to understand stories because it knew about scripts, PAM was envisioned as a program that could understand stories about situations for which no ready-made script exists. PAM would know about plans and goals. These, Schank theorized, are what enable people to function in situations they've never encountered before. (1984, 51)

Prior to Wilensky, work in planning had, like the theorem-solving programs of Newell and Simon, addressed only an aspect of planning, such as problem-solving, or was confined to "the area of 'robot planning'—programming robots to move blocks around" (ibid.).

The business of "programming robots to move blocks around" entails providing the program employed by a robot's computer with detailed knowledge about an environment, as well as about plans and goals. Michie considers the level of difficulty of programming such knowledge to be equivalent to that associated with the language understanding task. A robot in the most elementary of environments is, he explains, not unlike a child in a playpen with a few commonplace objects; it is up to the machine's programmer to provide it with abilities equivalent to those the child employs "to come to terms with,



and mentally organise, not only the visual and mechanical properties of material objects, but also the basic laws of nature, such as gravity, which determine their interaction" (Michie 1974, 103). A robot, with its associated paraphernalia—cameras, moveable platforms, and manipulators—is incidental to this task; "we are concerned," Michie explains,

> with the use we can make of a robot in order to develop *theories* of "computer understanding" of the real world. The acid test of such theories is provided by the robot itself. If it can be got to *understand* what is involved in operations such as "find," "fetch," "build," "tidy," and so on; then, and only then, can we program it to do these things. (ibid., 104)

In the early 1970s, at the Stanford Research Institute, formerly associated with Stanford University but now an independent research organization known as SRI International, a working model of a robot that "understood" a highly constrained environment sufficiently well to formulate and carry out simple plans within it was built. Nils Nilsson directed the project that resulted in the creation of Shakey, whose name derived, according to McCorduck, from the observation by one of the project staff that the mobile robot shook a great deal as it moved about (1979, 230).

Shakey was radio-controlled, stood about the height of a man, had no "hand," and could not climb. Michie proffers a report by Stephen Coles that describes a task-oriented situation in which Shakey was placed and recounts sketchily the strategy for enabling the robot to carry out the task.

> [Shakey] is in a room in which a box rests on a platform. The platform is too high for the robot's wheels to mount. Somewhere in the room is a ramp. The robot's task is to cause the box to be on the floor. . . . The capability to reason out the solution from first principles is programmed into the system. The key techniques are derived from mathematical logic. The trick is to dress up the formulation of a plan of action so that it looks exactly like the task of proving a theorem in some logical calculus. The kind of theorem the machine tries to prove is one which asserts that "a possible state of the world exists in which the box is on the floor." It is possible to arrange that, as a side-effect of a successful proof, a chain of actions is produced for bringing about the desired state of the world. (1974, 108)

Coles' report also provides a "robot's-eye-view" of how Shakey goes about completing the task; it "sets out," he explains, "the stream of 'thoughts' . . . which go through the robot's 'brain' (. . . the program running in Stanford's SDS940 computer) in the form of the following informal English translation:"

> My first subtask is . . . to move the ramp over to the platform and align it properly. To do this, I must first see it. To do this, I must first go to the place where, if I looked in the right direction, I might see it. This sets up the subsubtask of computing the coordinates of a desirable vantage point in the room based on my approximate knowledge of where the ramp is. . . . Next I have the problem of getting to the vantage point. Can I go directly, or will I have to plan a journey around obstacles? Will I be required to travel through unknown territory to get there if I go by an optimal trajectory; and, if so, what weight should I give to avoiding this unknown territory? When I get there, I will have to turn myself, and tilt the television camera to an appropriate angle, then take a picture in. Will I see a ramp? The whole ramp? Nothing but the ramp? Do I need to make a correction for depth perception? (ibid., 109-110)

"Shakey," according to McCorduck,

> taught its creators some surprising things. Perhaps one of the most important had to do with that elusive property of generality, for Shakey showed that you could not, for example, take a graph-searching algorithm from a chess program and hand-printed-character-recognizing algorithm from a vision program and, having attached them together, expect the robot to understand the world. . . . There are serious questions about the interaction between knowledge in different domains. . . . [and] with uncertainty in a complicated world. (1979, 232)

Terry Winograd, a doctoral candidate at MIT in the early 1970s, created a "bodiless" robot in software that he cryptically called SHRDLU, the concatenation of the latter six of the twelve most frequently used letters in the English alphabet. "SHRDLU," Mishkoff explains, "works in an imaginary limited domain called the 'blocks world,' consisting of blocks of various sizes and shapes" (1985, 95).

Modularity of design is a key facet of expert systems; they typically comprise a user interface, a knowledge base, and an inference engine which might be likened respectively to the door to a library, its collection, and the supporting mechanism by which it manages the ordering of, and access to, that collection.

"Early vision research in the late 1960s," McCorduck explains,

> had concentrated on the blocks world, the cubes and pyramids that robot arms moved around on command. To understand such a simple universe seemed . . . a summer's project. It wasn't. The problems were very hard . . . and depended not only on understanding image processing, but also on the predispositions of the observer, how far that system—human or robot—understood limits, and how it represented its knowledge internally. (1979, 258)

The question had been raised, McCorduck explains, of whether "intelligence—understanding both what one is told and what one can see for oneself—[might] turn . . . out to be the use of specific kinds of knowledge in a given context" (ibid., 259). Winograd's thesis, she notes, was one of a number of tests that sought to respond to it. Winograd suspected that programming methods, such as iteration, procedure, and recursion, had counterparts in human cognition, and he patterned SHRDLU's knowledge accordingly, arranging for the program's equivalents of reasoning, semantic analysis, and parsing to be carried out, as they are in humans, concurrently. "This simultaneous attack on several fronts," McCorduck notes, "saved SHRDLU from the embar-

rassing dead ends toward which semantic ambiguities had always led programs in the past, because at each stage of the understanding process, each bit of expertise made a contribution" (ibid.).

In deference to acknowledged limitations in the techniques available to him, Winograd, in order to provide his "robot" with sufficiently deep knowledge about its environment, severely circumscribed the world it would have to deal with. Further, by abstracting his blocks world, representing the totality of it—blocks, tabletop, arm, dialogue—on a video display, he sidestepped the mechanical engineering problems that would have attended a real world implementation.

SHRDLU, through the intermediary of a teletype, could be instructed to do things in its world, such as repositioning and stacking blocks; interrogated about what it had done and about the states of particular blocks; and, to some extent, even account for its reasoning.

> "it," as well as using deduction to decide which block it is holding. It inquires for clarification: "By 'it' I assume you mean the block that is taller than the one I am holding." There are two possible referents for "it:" the one selected, and "the one you are holding." The program has heuristics, rules of thumb, for making such choices.
>
> The human can ask abstract questions, such as, "Can the table pick up blocks?" which the program answers in the negative, for it knows as a fact that picking up must be done by animate objects, and the table is not animate. Or perhaps the human would like some history: "Had you touched any pyramid before you put the green one on the little cube?" "Yes," the program replies, "the green one," for it had deduced from its memory which event is being referred to and can relate other events to it in time. . . . The human says, "Does a steeple—" "Sorry," the program interrupts, "I don't know the word steeple." "A steeple is a stack that contains two green cubes and a pyramid," the human explains, and that definition is added to the program's knowledge in a form compatible with all the words it already knows. "I understand," it says, and in a limited but real sense it does. (ibid., 260)

The separation of inference engine from knowledge base gives the former wider application; unplugging one knowledge base from a generalized inference engine and plugging in a different one yields a consultation system for another problem domain for the labor of creating the knowledge base alone.

McCorduck relates representative examples of interaction between SHRDLU and a human.

> "Pick up a big red block," the human participant types onto the teletype. "OK," the computer answers back, and a linear "arm" moves across the television screen, picks up a green block on top of the red one, finds a space on the table to store the green block, then picks up the big red block. "Grasp the pyramid," says the human. "I don't know which pyramid you mean," the machine replies, for there are three pyramids in the scene, and the program knows that phrases beginning with the article "the" are intended to refer to a specific object the speaker has in mind. "Find a block that is taller than the one you are holding and put it into the box," the human says, and the system must figure out what is meant by "one" and

Gerald Sussman subsequently taught a program called HACKER, which he built in the image and likeness of SHRDLU, some new tricks. HACKER was provided with the ability to stop mid-course when it was stumped, analyze the subgoals that comprised its attempted solution, and try to detect and resolve conflicts among them—it could, in effect, debug the reasoning of which its predecessor was capable.

But SHRDLU did more than spawn more clever successors; it suggested something, McCorduck explains,

> that designers of other kinds of artificial intelligences, namely intelligent assistants, had simultaneously arrived at in their attempts to build programs that performed at the level of human experts in a given field. It was that humans operated successfully not by using powerful, underlying general rules, but rather by using a large amount of detailed knowledge, organized in special ways. This was the factor that distinguished the specialist from the amateur. (ibid., 262)

McCorduck describes an "intelligent assistant" as a program whose extensive knowledge in a narrow but difficult task domain is used to relieve a human expert of some of the taxing but essential parts of a particular job. Perhaps the earliest of these was DENDRAL, a program capable of inferring molecular hypotheses from chemical data. Provided with a chemical formula of a compound and data from mass spectrometer analyses, the program proffers from the vast number possible the most likely arrangements of the atoms in each molecule (Michie and Johnston [1984] 1985, 49). The multidisciplinary effort that yielded DENDRAL, which began at Stanford in 1965, included Edward Feigenbaum, who had been a graduate student under Simon at Carnegie-Mellon, Joshua Lederberg, a professor of genetics and Nobel laureate, and Carl Djerassi, a physical chemist (McCorduck and Feigenbaum 1983, 61-62).

Research into DENDRAL-like systems continued at Stanford; in the early 1970s, under the aegis of the Heuristic Programming Project, work on what Newell would later call the "grandaddy" of all expert systems was begun by Bruce Buchanan of the Stanford Department of Computer Science and Edward Shortliffe of the Stanford Medical Center (Mishkoff 1985, 41). Their prototype "expert system," called MYCIN, brings to bear on the diagnosis of infectious diseases and recommendation of appropriate therapies the collective expertise of many physicians. [A related article, in the "Newsworthy" section of this issue, describes how a project team at Harvard is collecting and incorporating into an expert system domain-specific knowledge from human experts in the field of law.]

Newell's observation regarding the stature of MYCIN is well taken; open it up and one can detect the

Expert systems and their component pieces, inference engines, knowledge bases, and natural language processors, are the current commercial products of artificial intelligence.

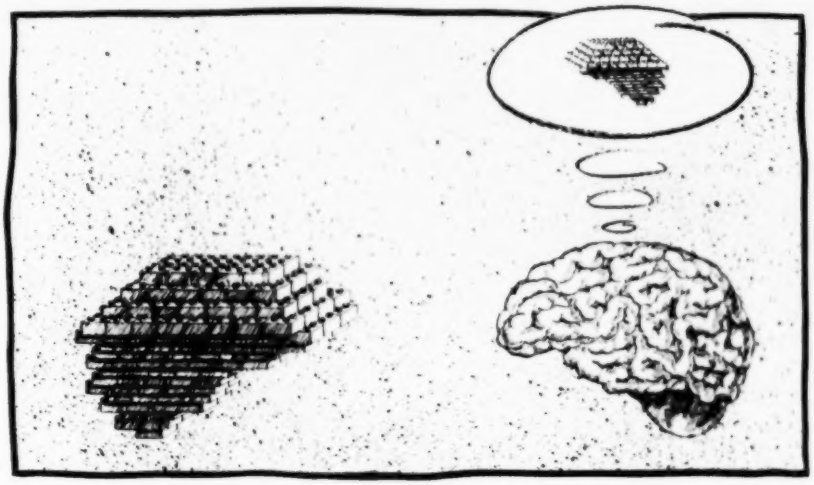
elemental fragrances whose adept blending is the essence of every successful, present day, expert system. The components of this essence include several facets of problem identification, conversational abilities, accountability, modularity of design, and reliability in operation.

MYCIN's problem definition is exemplary. It was conceived to operate in a vital domain, medicine, on real problems associated with diagnosis of infectious diseases, such as overuse, misuse, and side-effects of antibiotics, effects of antibiotics used in combination, and maldistribution of expertise. Additionally, its problem domain was constrained sufficiently to make it tractable to existing limitations in expert system design techniques.

Physician-program interaction is conversational; the former provides the latter with background information that it could not infer, such as data drawn from patient history and laboratory tests, and responds, typically with YES or NO replies, to questions generated by the program as it reasons toward possible diagnoses and therapies. "This style of interaction," explain Winston and Prendergast, "is similar to what goes on in hospitals. There are specialists on hospital staffs available for consultation whenever a less-specialized physician wants expert advice. MYCIN's authors designed the program to fit into the existing pattern of seeking and providing consultative help" (1984, 31).

Technically speaking, MYCIN's conversational skills derive from its *user interface*. "The communication performed by a user interface," Mishkoff explains, "is bidirectional. At the simplest level, you must be able to describe your problem to the expert system, and the system must be able to respond with its recommendation" (1985, 57). Because expert systems are frequently intended to be used by individuals who are generally unfamiliar with computers, English language exchanges are highly desirable; the techniques for implementing such interfaces, Mishkoff notes, derive from a related discipline called *natural language processing*.

"In practice," Mishkoff adds, "a user interface generally is expected to perform additional functions. You may want to ask the system to explain its 'reasoning,' for example, or the system may request additional information about the problem" (ibid).



What MYCIN "knows" is a set of "if-then" rules of a form such that "if certain conditions obtain, then there is suggestive evidence that—." A physician or medical assistant who wants the program to account for its reasoning

> can type RULE, and this will cause the machine to display the rule that it is currently evaluating. This gives the user an idea of what is happening. For a more detailed explanation, the user asks WHY and gets a summary of the logical process that is being worked through. . . . The explanations are composed from the English text that is held in the machine in conjunction with each rule; the rules that the machine actually operates are held internally in advice language. (Michie and Johnston 1985, 44-45)

Modularity of design is another key facet of expert systems. The user interface might be thought of as a door to the expert system proper, which, in turn might be thought of as a library with its associated support structure. In expert system parlance, these latter constitute the *knowledge base* and the *inference engine*, respectively.

Like the books on a library's shelves, the knowledge base is the collection of all the information available to the system. Mishkoff makes a useful distinction between the more venerable concept of computer *database* and the knowledge base of an expert system. Although storage and retrieval of information in a particular domain are implicit in both, the knowledge in a database, he explains, "is only declarative (factual) knowledge" (1985, 54), whereas "a knowledge base contains both declarative knowledge (facts about objects, events, and situations) and procedural knowledge (information about courses of action") (ibid., 56).

As the knowledge base is to a library's collection, so the inference engine is to its control structure, that is, all of the mechanisms by which the institution acquires, orders, integrates, and makes accessible, the constituent pieces of its collection.

Randall Davis, an early contributor to the MYCIN project and the developer of TEIRESIAS, a tool for knowledge acquisition, describes the modular design of the former in *The*

AI Business. MYCIN's architecture, he explains, set a pattern for expert systems. It comprises a knowledge base whose nearly five hundred rules represent most of what is known about infectious disease diagnosis and therapy, and an inference engine that attends to finding, extracting, and applying that knowledge (Winston and Prendergast 1984, 33).

The if-then form of the rules in MYCIN's knowledge base renders individual rules simple and understandable even out of context, unlike traditional program code which usually makes very little sense when read a few lines at a time. Davis suggests that successful applications for MYCIN-like technology require a domain that can be so chopped into a few hundred or thousand rule-sized chunks, each of which constitutes a separate, distinct, and comprehensible piece of information (ibid., 34-35).

MYCIN's inference engine, given the name of a suspect organism, locates and extracts from the knowledge base all of the rules that make conclusions about identity and works recursively through these until it exhausts them, at which point it asks the physician a question.

Modularity has yielded another benefit—generalizability of the inference engine. Davis explains that this separation of inference engine from knowledge base gives the former wider application; unplugging one knowledge base from a generalized inference engine and plugging in a different one yields a consultation system for another problem domain for the labor of creating the knowledge system alone (ibid., 34). By way of example,

> to help construct expert systems in other spheres, the disease rules were taken out of MYCIN, leaving a general-purpose system called Essential MYCIN, or EMYCIN. Using this, a quite separate expert system was constructed to diagnose lung disease. . . . PUFF. . . . takes data from a spirometer, a machine that measures air flow rates and volumes as the patient breathes in and out of a tube. . . . asks questions about the patient's history—how many cigarettes a day and so on—and then, using a knowledge base of about a hundred rules, produces a detailed description of the patient's apparent condition and a diagnosis of disease. (Michie and Johnston [1984] 1985, 47)

The difficulties of trying to get programmed machines to act in the manner to which we are accustomed to perceiving one another act provide many amusing anecdotes.

With regard to reliability, Michie and Johnston add that "the doctors who check . . . PUFF's reports sign 85 percent of them unchanged," and of MYCIN they report that a panel of experts assessing the anonymous prescriptions for ten difficult cases made by MYCIN and nine physicians assigned the highest score to the program (ibid., 46-47).

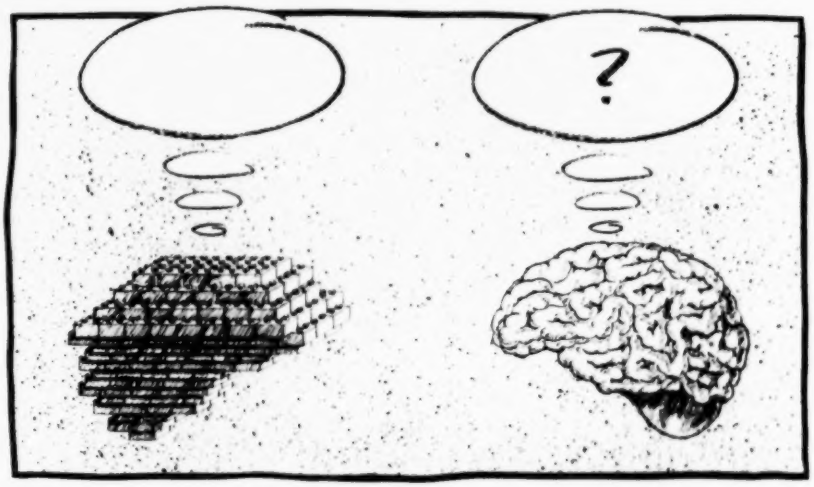
The efficacy of expert systems continues to be demonstrated by successor systems in a growing number of fields. Michie and Johnston recount several, among them, the sixteen hundred rule Prospector system, developed by SRI, International, which has elicited interest from mining corporations and from government agencies concerned with energy conservation and exploitation of natural resources. "Prospector," they explain,

> takes in geological data about the rocks and ores observed in a given area of land, levels of erosion, and so forth, and produces forecasts of what valuable minerals might be found there. . . . Prospector uses an inference network of hypotheses, which it tests. . . . by asking appropriate questions, and. . . . [which] can also be linked by "context constraints." . . . [that] say in effect: "Don't even consider hypothesis A unless the probability of hypothesis B falls within such and such a range."
>
> Prospector can also reason intelligently about the information it holds. For instance, using a separate "taxonomy network" it can know that pyrites are a type of sulfide, so if it already has some information about pyrites it can avoid asking redundant questions about sulfides. Similarly, if the user tells it, "There are pyrites present," and then "There are no sulfides," the machine can say, "Hold on There's a mistake there." (ibid., 47-49).

In the field of genetics, Michie and Johnston cite Molgen,

> an expert system for helping geneticists plan experiments that involve the manipulation of DNA, the basic carrier of heredity. This entails cutting up the very long molecules of DNA, joining the pieces together, inserting new materials in . . . myriad different possible places, and keeping track of the biological consequences and the chemical tools and instrumentation required. Molgen produces a plan, often thousands of steps long, to organize . . . these. . . . [and] has been used effectively in a number of experiments, including . . . one involving gene cloning in rat insulin. (ibid., 49-50)

What is perhaps the first commercially viable expert system to achieve routine use began as a joint development project between Digital Equipment Corporation (DEC) and Carnegie-Mellon University in



1978. Two years later, the system, designed to verify configurations of DEC VAX computers, matriculated to the firm's industrial environment. XCON, so dubbed for its

Like agricultural robots, expert systems are engineered to labor tirelessly and reliably in fields large and small, and differently planted, but in every case bounded.

expert configuration skills, brings to the task of reviewing the plethora of possible configurations of the firm's VAX computers approximately forty-five hundred rules and a database of more than nine thousand components. The system verifies configuration feasibility as well as completeness, checking components for adequacy and compatibility, evaluating memory size, and making determinations relative to the type and length of connecting cables. The system "knows" about the sizes and shapes of the cabinets the firm uses, even to the location of the slots that components fit into.

XCON addresses a very real problem for DEC, a hallmark of whose computer sales is diversity. In the first two years of production use, XCON analyzed more than twenty thousand unique orders at an average time of two minutes per order as opposed to from twenty-five to forty minutes for expert technical editors, and at a level of accuracy of more than ninety-five percent as opposed to between sixty-five and seventy percent for the editors.

DEC's success with XCON has led to the development of two complementary expert systems, XSEL, a highly interactive system that, imbued with all of XCON's knowledge and more, will enable a DEC sales person to prepare quotes on a portable terminal at a customer site, and XSITE, a system capable of producing, for a given configuration, a floor plan depicting the arrangement and connections among system components, and of specifying environmental requirements, such as air conditioning and electrical power.

XCON is also known as R1, the latter appellation apparently deriving from the head of the project team that developed it, Carnegie-Mellon professor John McDermott, who reportedly uttered the the following variation on a classic phrase: "Three years ago I wanted to be a knowledge engineer, and today I are one" (Winston and Prendergast 1984, 41). "Knowledge engineer" has only recently entered the vernacular, but the notion implicit in the term was expressed in print as early as the late 1960s in Jaki's book; discussing the simulation of thinking operations for useful purposes, Jaki referred to "thought engineering" done by "information engineers," whose training, he noted, "is not geared to developing appreciation for basic scientific and philosophical questions" (1969, 248).

Although "knowledge engineer" has only recently entered the vernacular, the notion implicit in the term was expressed in print as early as the late 1960s in a book by Stanley L Jaki.

MacKay once referred to the computer as the "honest hodman par excellence" (ibid., 242). The phrase is equally apropos of the expert system. Like agricultural robots, these systems are engineered to labor tirelessly and reliably in fields large and small, and differently planted, but in every case bounded.

Expert systems and their component pieces, inference engines and natural language processors, are the current commercial products of artificial intelligence. Work going on at other unversities and, increasingly, in corporations and small specialty firms, will yield others. Mishkoff cites a host of research efforts—variously involving industry, academe, and academic-industrial alliances—in such areas as automatic programming, self-contained vision systems, planning systems for robots, speech recognition and generation systems, and "intelligent" tools for developing expert system components (1985, 42-43).

It may be that the hype accompanying the commercialization of artificial intelligence will one day describe actual system characteristics and capabilities; the progeny of artificial intelligence research a decade hence might be as unimaginable today as expert systems were a few decades earlier.

Douglas Hofstadter, speculating on whether emotions will ever be explicitly programmed into a machine, suggests that the notion

> is ridiculous. Any direct simulation of emotions . . . cannot approach the complexity of human emotions, which arise indirectly from the organization of our minds. Programs or machines will acquire emotions in the same way: as by-products of their structure, of the way in which they are organized—not by direct programming. Thus, for example, nobody will write a "falling-in-love" subroutine. (1979, 677).

Except, perhaps, a pair of graduate students at the University of California at Berkeley. Peter Norvig and Joe Faletti are subjects in Rose's humanistic study of artificial intelligence research at Berkeley. They were, while he was writing about them, trying to marry a frame-based memory system to a story-understander program called PAM and a problem-solving program named PANDORA; the object?—the courtship of John and Mary.

The composite program, called PAMELA, was intended to be able to make distinctions about love and to entertain a flood of associations at certain suggestions. Frames possessed both characteristics and defining features, which the designers defined as things one might expect to find in a frame, and things that had to be there, respectively. For the Romantic Love frame, for example, a defining characteristic is that a man and a woman share an attitude called "love;" a characteristic might be that the man and the woman are both adults. The flood of

associations the two wanted PAMELA to make were the kinds of connections that are made unconsciously by humans, as between marriage and wedding rings, honeymoons, and baby carriages (Rose 1984, 70-71, 79-80).

Faletti had earlier attempted to imbue PANDORA, the problem solving component of PAMELA, with a knowledge of the weather and its consequences. As the "raincoat program," PANDORA was provided with a desire to learn more about the world upon waking. It knew that one could learn about the world from newspapers, and that a newspaper was to be found on its front lawn. PANDORA the raincoat program also had an understanding of the states of being wet and dry, a preference for the latter, and the knowledge that rain resulted in getting wet and that a raincoat could help preserve the state of being dry.

In its infancy, PANDORA learned to put its raincoat on—and on, and on. It fell into what is known in programming jargon as a loop. Though its programmers remedied this situation, getting PANDORA to put its raincoat on but once and then retrieve the newspaper, their success was tempered by some confusion on its part regarding indoors and outdoors; having determined that it was raining, the program put its raincoat on and retrieved the newspaper but, alas, elected to read it out of doors (ibid., 59, 62-68, 177-180).

PAMELA, too, was to have limited success. Told that "John loves Mary," PAMELA appropriately invoked the Romantic Love frame which put it in the state of knowing that there was associated with John a mental object which was an attitude of love for Mary. Where PAMELA bogged down was in the Ask frame, which was to be full of knowledge about asking questions and getting answers. Poor PAMELA interpreted one of Faletti's questions as the answer to it with results little less devastating than can occur when one human lover misconstrues another's question for an answer (ibid., 81-83).

The difficulties of trying to get programmed machines to act in the manner to which we are accustomed to perceiving one another act provide many such amusing anecdotes.

Margaret Boden suggests that the advent of expert systems wants a new kind of computer literacy, specifically, the sharing of knowledge about the nature of limitations that can apply to such systems with not only those who actually use the programs but also those who might be affected by their use.

The bases of PAMELA and PANDORA are the understanding and observation of human mental processes as these are experienced by the programs' human progenitors. That some might prefer deeper access to such processes is suggested to Jaki by the observation of one author that "it is undesirable and illegal to put a human being in a completely non-operating state ... and once this is done, the human being can never be returned to an operating state—he is dead ..." (1969, 255). The missing text implied by the elipses are Jaki's own observations: relative to the first part of the sentence, that "old fashioned humanists would have said 'to kill him,'" and of the concluding qualification, that "this explanatory remark is probably added for the benefit of those who are no longer familiar with 'anthropomorphic' notions such as death" (ibid.). Jaki's concern is for what could occur were we to give over too fully to the tendency that has persisted throughout the history of artificial intelligence—to impart, as Hobbes did, an artificial life to our automata, while at the same time reducing our own life to the most elementary of mechanical terms.

Michie and Johnston extend Jaki's concern to the artificial automata that might one day contain some representative characteristics of human organisms. Noting, under the heading "Killing an expert system," that "animals often have to be 'put down' for practical reasons, and so will robots," they posit that

> an interesting question arises concerning expert systems. One of these constructed with the cooperation of a particular human expert can capture his skills in such faithful detail that colleagues and friends interacting with the system may recognize his personal style of thought. Long after he is dead, they could respond to the foibles and fancies of the intellectual companion they knew. This would be for him a kind of immortality, more vivid and direct than authorship of books, or passive relics such as photographs and tape recordings, since it would include fragments of responsive behavior. How should we regard the wiping out of such a system irretrievably? It would surely be reprehensible, in the same way as was the burning of the great library of Alexandria. (Michie and Johnston [1984] 1985, 222)

Would it be? How is one to judge? Perhaps it is too early for such conjecture. Perhaps. But very probably it is not too early to reflect on what might seem by comparison more pedestrian issues related to the anticipated proliferation of artificial intelligence-based products, and perhaps our ability to deal with more radical possibilities will be abetted by this earlier questioning.

Margaret Boden, of the University of Sussex in Great Britain, suggests that a new kind of computer literacy is wanted. To the notion that edu-

Boden contends that it is always reasonable, in principle, to question a computer program and the models it employs, and encourages users to not "take the answers, the judgements, the advice given by computer programs to be, if not absolutely objective, at least somehow more objective than the decisions of mere human beings." (Bernold and Albers, 225)

cation of users is not going to be necessary "because the whole point of AI is that with these natural language interfaces, and reasoning power, and interactive facilities, the user is not going to have to know about computers," Boden responds that,

> certainly, they are not going to have to know about the way the thing is built, and they are not going to have to know about the details of the programming languages. But there are certain things that they *are* going to have to know, and which I think are going to have to be taught. (Bernold and Albers 1985, 224)

One of the things Boden suggests needs to be made clear is the nature of the limitations that can apply to such programs, and it is important, she believes, that such knowledge be shared not only with those who actually use the programs, but also with those who might be affected by their use. Such limitations, she proposes, might best be flagged by those who write and market the programs.

The other thing that Boden contends should be kept in mind by all—programmers, non-programmers, users, and non-users, alike—is that it is always reasonable, in principle, to question a computer program. "You can do this," she explains,

> on three levels: You can question its data. That is to say you can ask: "Does it have all the necessary data, all the relevant data?" (because it may not). And you can ask: "Are all the data which it has actually true?" (which they may not be). In the same way you can ask about a human being: "Do they know all the facts, and are all their beliefs which they take to be relevant, in fact true beliefs?". . . . Similarly, you can always in principle question the rules of inference that the thing is

Cognitive science and artificial intelligence

"The progress of science tends to be limited by the tools available to it." This statement may seem obvious and therefore unnecessary, but the problem runs deeper than the simple assumption that most scientists tend to formulate problems in such a way that they can be solved using tools and instruments available to them. This process can slow progress for long periods of time when problem-definition, thus circumscribed, becomes accepted wisdom in a scientific community, affecting how scientists perceive their work—forming their "models of perception."

The study of the human mind has suffered this disease more than most fields of inquiry. Probably the most ancient, and still pervasive, limitation of our view of the mind is that what we perceive as our mind is what we remember of our conscious thought. This has been true since the unrecorded beginnings of human introspection. It is still true. In the era from the time of the ancient Greek philosophers to the present, this model of perception has persuaded us that the mind is an engine of logical thought. Here is how that notion arises—and persists.

Introspection tends to concern itself with the mental process that is usually observed by this method—conscious thought which is remembered and reperceived, a selection and modification process. It is undeniably evident that the conscious mind is capable of symbol-manipulation, computation and ratiocination, the dominant conscious mental activities of the philosophers and scientists who were observing their own minds at work. It was—and still is—likely, perhaps inevitable, that these thinkers would believe that logic or reason is the primary function of the mind.

This view of the mind persists to the present day, reinforced by, and reinforcing, the traditional western view of man as a rational being—an essential conceptual underpinning of western social philosophy and democratic institutions. This cultural bias exerts a powerful, almost irresistible influence on our perception of the human mind.

The development of the digital computer has further reinforced this perception. The first computers were referred to as "mechanical brains," then "electronic brains." These early characterizations of the computer as functionally identical to the human mind arise from the deeply-rooted consensual notion of the mind as the seat of reason, a logic machine, more complex than, but essentially the same as, a man-made logic machine, the digital computer. And as the computer has become more powerful, it has been used increasingly for analysis and decision making. Such programs now fall into a class known as *artificial intelligence*, a term which to many means artificial *human* intelligence.

Thus the digital computer has become the latest model of perception of the human mind and is shaping much of the work done in the developing field known as *cognitive science*. Some who work in both fields—artificial intelligence and cognitive science—believe that these two disciplines will ultimately converge, and that, some day, a computer will faithfully replicate human mental activity.

Other disagree, arguing that logic, computation and ratiocination are functions the human mind can learn to perform (albeit slowly and imperfectly), but that they do not constitute basic brain function. They maintain that whereas the circuitry of a serial digital computer makes it a logic machine and nothing else, the logical operations that the human mind sometimes performs *are but one among many mental operations*, all of which are produced by a *neural network, the organization of which bears no resemblance to computer circuitry.*

Toilers in the field of artificial intelligence will probably not be deterred by these obstacles. It is well that they should press on, for their work contributes much of value to the development of computer science, which continues to yield highly useful results:

- *More useful software* for the control and management of the processes of human enterprise, programs that can "learn" from shifting patterns and reprogram to cope.
- *More powerful hardware:* cheap, virtually unlimited internal memory; multiple-channel simultaneous processing with cross-talk; sensory input devices to collect data directly from processes in real time; output devices directly connected to and controlling processes.
- *Integration of computers with processes* on a larger scale. As these systems become more complex and independent of human intervention, we may start thinking of them as "organisms."

Shall we then have *artificial intelligence*? Certainly. We have it now. Later systems will be so smart as to appear superhuman—and so they will be in some respects. And with their self-programming ability to adjust to changes in their environment, they will appear to emulate the human organism. Some will even, at times, run amok, prompting some of us to attribute to them human characteristics such as madness or evil intent.

But will they be models of the mind? They will certainly be *used* as models of the mind, but if this use is based on the notion that the mind is fundamentally and primarily a computational organ, they will teach us something about cognitive processes, but not much about the mind.

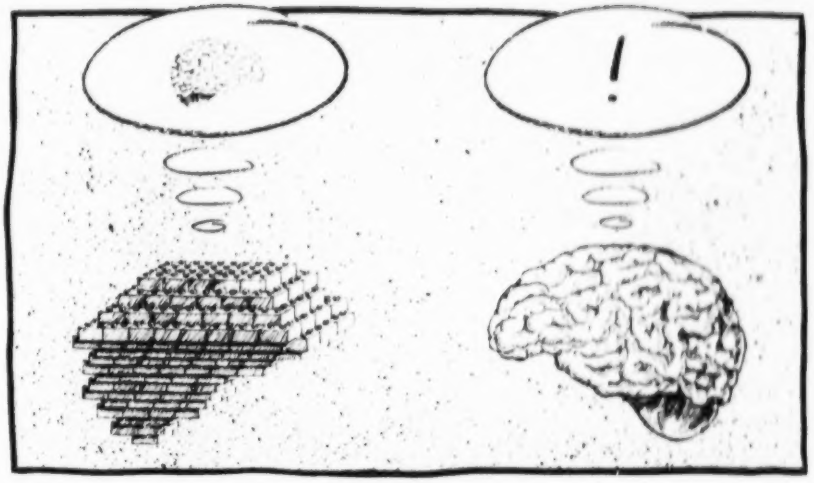
Donald S. Bradshaw
Harvard College A.B. 1944

> using, just as you can in principle question the rules of inference that the human being is using. . . . And thirdly, in principle, it is always possible to question the values, or decision criteria, which are built into the program—just as you can with human beings. What you think is "best" may not be what the programmer assumed was "best." (ibid.)

Finally, Boden points out that artificial intelligence programs are basically knowledge systems that comprise models of the world that are analogous to those humans carry around with them, and that we have an obligation to question these models as well. Boden contends that many people do not, that "they take the answers, the judgements, the advice given by computer programs to be, if not absolutely objective, at least somehow more objective than the decisions of mere human beings" (ibid., 225).

So we come full circle; if we heed Boden's advice, we are granting that artificial intelligence systems are just what they are, and treating them accordingly. Mr. Turtle never gave up, however. If only in reruns on UHF television stations, he even today pleads with Mr. Wizard to let him be something other than what he is. Artificial intelligence programs will undoubtedly become something other than what they are presently. Very probably they will become more than they are now. Will we recognize them for what they've become, for all that they become? Or will we impose upon our own minds the kinds of boundaries that the artificial intelligence programs of today admit, perforce.

While at Berkeley, Rose observed taped on the door of the fifth floor office of the artificial intelligence group in Evans Hall a cartoon clipped from the pages of the science fiction magazine *Beyond*. "The comic strip," he relates, "was called 'Boid' . . . [It read] 'Think yer smart, don't ya?' said a bird to a computer. 'Mebbie you can compute numbahs faster than us livin' creatures . . . But can you feel? Can you laff & cry, an' hate, an' love? Na, computer, fer all yer smarts, you'll never replace us.' Then the bird flew away—and the computer said, 'Oh, well, I can dream, can't I?'" (Rose 1984, 61). *J.S.*



Bibliography

Artificial intelligence: a long and winding road

Books

Bernold, Thomas, and Günter Albers, eds. *Artificial Intelligence: Towards Practical Application.* New York: Elsevier Science Publishing Company, Inc., 1985.

Dreyfus, Hubert L. *What Computers Can't Do.* New York: Harper & Row, 1979.

Harvard University. *The Annals of the Computation Laboratory of Harvard University,* vol. 1. Cambridge: Harvard University Press, 1946.

Hofstadter, Douglas R. *Gödel, Escher, Bach: An Eternal Golden Braid.* New York: Basic Books, Inc., 1979. Reprint. New York: Vintage Books, 1980.

Jaki, Stanley L. *Brain, Mind and Computers.* New York: Herder and Herder, 1969.

Lattimore, Richard. *Homer's Iliad.* Chicago: University of Chicago Press, 1951.

McCorduck, Pamela. *Machines Who Think.* San Francisco: W. H. Freeman and Company, 1979.

McCorduck, Pamela, and Edward A. Feigenbaum. *The Fifth Generation.* Reading, Massachusetts: Addison-Wesley, 1983.

Michie, Donald. *On Machine Intelligence.* New York: John Wiley & Sons, 1974.

Michie, Donald, and Rory Johnston. *The Knowldege Machine.* First published in a slightly different form as *The Creative Computer.* Harmondsworth, Middlesex, England: Penguin Books, Ltd., 1984. New York: William Morrow & Company, Inc., 1985.

Mishkoff, Henry C. *Understanding Artificial Intelligence.* Dallas: Texas Instruments Information Publishing Center, 1985.

Rose, Frank. *Into the Heart of the Mind.* New York: Harper & Row, 1984.

Winston, Patrick H. *Artificial Intelligence.* Reading, Massachusetts: Addison-Wesley, [1981] 1985.

Winston, Patrick H., and Karen A. Prendergast. *The AI Business.* Cambridge: The MIT Press, 1984.

Articles

Kidder, Tracey. "Less (and More) Than Meets the Eye." *The New York Times Book Review,* 29 December 1985.

McCulloch, Warren, and Walter Pitts. "A Logical Calculus of the Ideas Immanent in Nervous Activity." *Bulletin of Mathematical Biophysics* 5 (1943), 115-133.

Oettinger, Anthony G. "Programming a Digital Computer to Learn." *Philosophical 7,* no. 43 (1952).

Calculating and searching

Books

Mishkoff, Henry C. *Understanding Artificial Intelligence.* Dallas: Texas Instruments Information Publishing Center, 1985.

Weiner, Philip P. *Leibnitz Selections.* New York: Charles Scribner's Sons, 1951.

Information Technology

CALENDAR

[*Information Technology Calendar* publishes listings of courses, lectures, seminars, and other events which have relevance to the subject of information technology. Listings are solicited from Harvard schools, departments, offices, and organizations which offer or sponsor such activities. Publication will be on a space-available basis.]

Courses

Word Processing Labs, Training, and Workshops (various packages for the DECmate II and IBM PC, including Samna Word III, Wordstar, and Lotus 1-2-3), classes begin weekly, Personnel Office Center for Training and Development, Memorial Hall, (Pat Stano, 495-4895).

Text to Type: An Introduction to Typography, Copyfitting, and Word Processing, April 5, Center for Lifelong Learning, (495-4973).

The Essentials of Computer Graphics: Current Developments, April 5, Center for Lifelong Learning, (495-4973).

Advanced N/I/TROFF (word processing under UNIX), begins April 14, Health Sciences Computing Facility, (732-1618).

Typesetting with PC TeX, April 14, Computing and Information Utilities, (495-3767).

Desktop Computer and IBM Mainframe Communications, begins April 18, Computing and Information Utilities, (495-3767).

Computers in Fund Raising, April 26, Center for Lifelong Learning, (495-4973).

Data Base Management on Microcomputers, April 26, Center for Lifelong Learning, (495-4973).

Word Processing Concepts, April 26, Center for Lifelong Learning, (495-4973).

Demonstration of MicroSoft Word on the Macintosh, April 28, Computing and Information Utilities, (495-3767).

An Introduction to Lotus 1-2-3 Spreadsheet/Financial Modeling, May 3, Center for Lifelong Learning, (495-4973).

The Computer Selection Process, May 3, Center for Lifelong Learning, (495-4973).

Communications Workshop: PC/FOCUS, May 5, Computing and Information Utilities, (495-3767).

Final Word and Perfect Writer, May 5, Computing and Information Utilities, (495-3767).

Communications Workshop: Hayes Smartcom II, May 7, Computing and Information Utilities, (495-3767).

Conferences

Integrating Computing into the Higher Education Curriculum, Drexel University, March 25-26, University of Washington, April 7-8, Northwestern University, May 12-13, Duke University, May 27-28, EDUCOM, P.O. Box 364, Princeton, New Jersey, (Mindy Weinberg, 609-734-1885).

Statistics, April 4, New England Regional Computing Program, 439 Washington Street, Braintree, Massachusetts 02184, (617-848-6494).

Teleconferences

(Unless otherwise noted, teleconferences are at the Gutman Conference Center, the Gutman Library; contact Dave Douglas, Video Services Center, 495-7597.)

Managing Resistant Ventrical Arrhythmias, April 16, 4-7:30pm, Network for Continuing Medical Education.

Incentive Compensation Systems, April 17, 1-4pm, American Hospital Association.

Introduction to Midazolan, April 22, 4-7:30pm, Sherman Fairchild Lecture Hall, Rausch Laboratories/Network for Continuing Medical Education.

Managing Viral Infections, April 30, 10:30am-1pm, Association of Hospital Television Networks.

A Passion for Excellence Within the Hospital: Featuring Tom Peters, May 8, 1-4pm, American Hospital Association.

To Be Announced, May 29, American Hospital Association.

To Be Announced, June 19, American Hospital Association.

Harvard University
Information Technology Quarterly
Office for Information Technology
1730 Cambridge Street
Cambridge, MA 02138

NON-PROFIT
ORGANIZATION
U.S. POSTAGE
PAID
BOSTON, MASS.
PERMIT NO. 1636

B. J VAUGHN
ERIC CLEARINGHOUSE ON INF.RES.
130 HUNTINGTON HALL
SYRACUSE UNIVERSITY
SYRACUSE,NY 13210